User's Guide

Publication number 16522-97007
June 2000

For Safety information, Warranties, and Regulatory information, see the pages behind the Index

Agilent Technologies 16522A 200-M Vectors/s Pattern Generator

The Agilent Technologies 16522A Pattern Generator

The Agilent Technologies 16522A pattern generator module is an expandable stimulus tool designed for the Agilent Technologies 16500B/C Logic Analysis System. It provides digital design teams the ability to emulate missing devices and to functionally test prototypes.

Key Characteristics

Output channels: 20 channels at 200 MHz clock; 40 channels at 100 MHz clock.

Memory depth: 258,048 vectors.

Logic levels (data pods): TTL, 3-state TTL/3.3v, 3-state TTL/CMOS, ECL terminated, ECL unterminated, and differential ECL (without pod).

Data inputs: 3-bit pattern - level sensing (clock pod).

Clock outputs: Synchronized to output data, delay of 11 ns in 9 steps (clock pod).

Clock input: DC to 200 MHz (clock pod).

Internal clock period: Programmable from 5 ns to 250 µs in a 1, 2, 2.5, 4, 5, 8 sequence.

External clock period (user supplied): DC to 200 MHz.

External clock duty cycle: 2 ns minimum high time.

Maximum number of "IF condition" blocks at 50 MHz clock: 1.

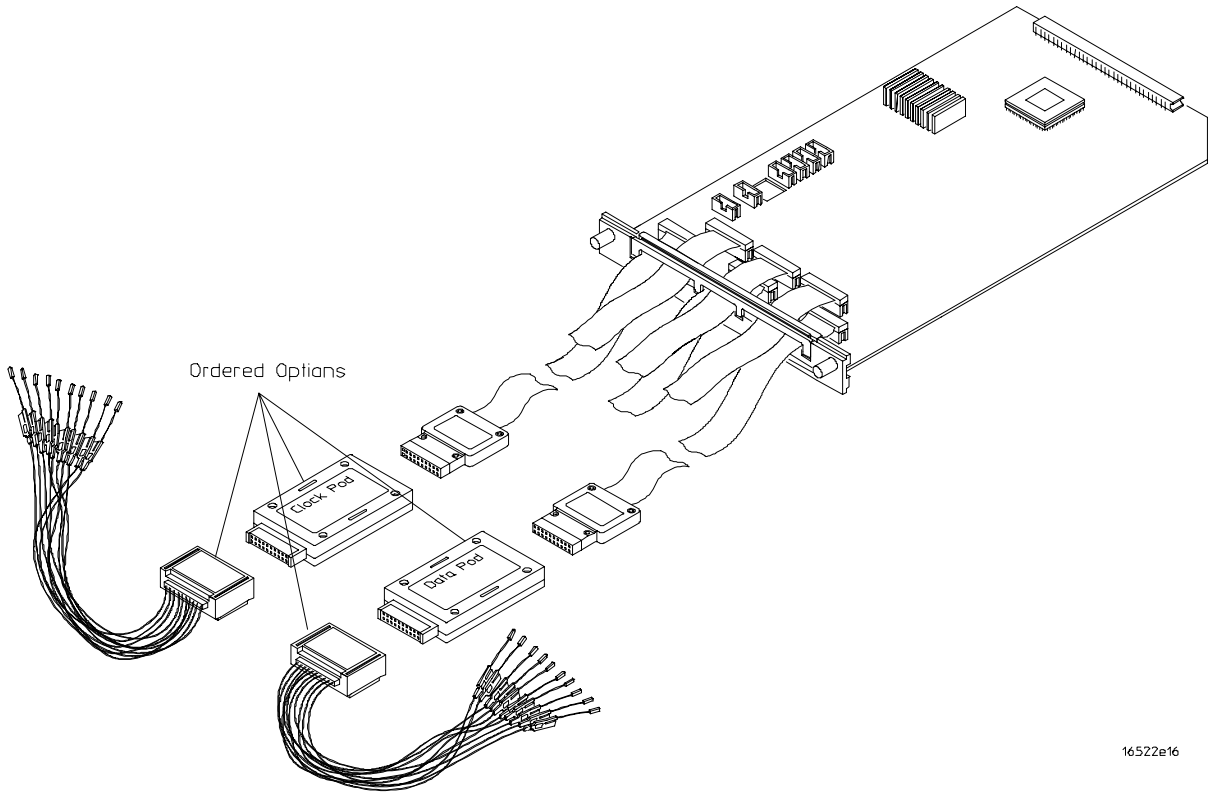
Maximum number of different macros: 100.

Maximum number of lines in a macro: 1024.

Maximum number of macro invocations: 1,000.

Maximum number of Repeat loop invocations: 1,000.

Maximum number of Wait event patterns: 4.



Agilent Technologies 16522A Pattern Generator

In This Book

This User's Guide contains all the information needed to operate, program, and service the Agilent Technologies 16522A Pattern Generator module.

Chapter 1 documents many common tasks that you will perform while using the pattern generator. Use this chapter as a quick reference to the product's operation.

Chapter 2 contains reference information on the functionality of the menu fields and product features. Use this chapter for additional information on feature usage and limitations.

Chapter 3 documents the programming command set used when controlling the pattern generator from a computer. Use this chapter for command reference information when writing your test verification programs.

Chapter 4 contains service procedures for verifying the product's performance, board-level troubleshooting, and replacement of the PC board and cables. Use this chapter if you suspect problems with the pattern generator.

Chapter 5 contains information on the installation of the pattern generator module and probe tip assemblies. Use this chapter to configure and install single and multi-card modules.

Chapter 6 documents how to load user-created pattern generator files as ASCII files into the pattern generator. Use ASCII file downloading if you want to create and load files by one of the remote communications interfaces or load by disk.

Contents

3 Programming the Agilent 16522A

1

Using the Agilent Technologies 16522A

Using the Agilent Technologies 16552A

This chapter provides instructions for using the Agilent Technologies 16522A to generate vectors and patterns for design and test environments. Two sections comprise this chapter:

- Setting up the proper configurations
- Building vectors and macros

Because the Agilent 16522A operates in much the same way other modules in the Agilent 16500B/C mainframe, discussions of general operation have been eliminated from these procedures. If you have questions of a general nature, refer to the User's Reference for the Agilent 16500.

See chapter 5 for installation instructions.

Setting Up the Proper Configurations

This section discusses setting up the configuration attributes and parameters of the pattern generator.

If you are reloading existing configurations or downloading ASCII vector files, refer to the Load operation in the disk drive menus of the System.

To set up the configuration

1 **From the Format menu, set the Vector Output Mode to either full or half channel.**

Make this selection first because clock frequencies and available channels are affected.

In half channel mode, only pods 1, 3, and 5 are used. Only four channels are available in pod 5.

2 **Set the Clock Source to either internal or external.**

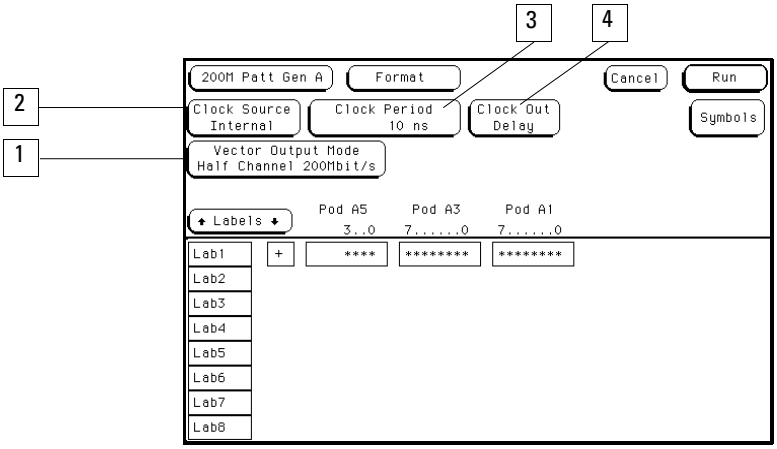
An external clock is used when you need a clock frequency that is not available internally, or if you need to drive the pattern generator timing with an external reference.

3 Set the Clock Period for an internal clock or the Clock Frequency for an external clock.

The external clock frequency information is required to select the appropriate operating mode. Operating the pattern generator at an external clock frequency higher than selected will result in erroneous operation.

4 Set the Clock Out Delay if a delay is needed.

Setting a delay is useful when using the clock out edge as a read strobe. If you do not set the Clock Out Delay, the value is uncalibrated.

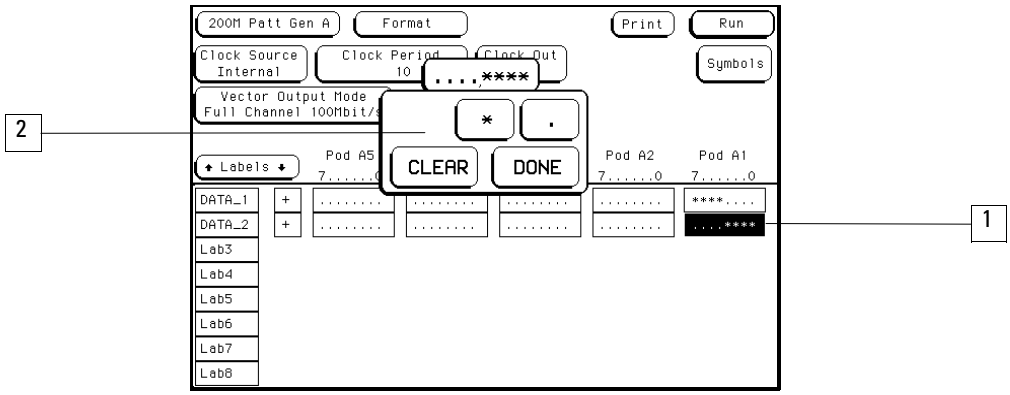


To build a label

When you build a label, you are grouping channels under a label name and mapping the selected channels to the probes on the associated pods. A label may contain up to 32 channels, however, a single channel cannot be used under more than one label.

For more information on using labels, refer to "Label Assignment" in the Common Module Operations section of the mainframe manual.

1 Select the label's channel assignment field.



2 Select the desired channels.
* (asterisk) = on

. (period) = off

Only the selected channels can output pattern generator signals.

3 Select Done.

Building Test Vectors

Once the pattern generator is configured, you will want to build programs to use in your test system. You build programs in the Sequence menu. If you have small program segments that are built from frequently used vectors, they can be built in the User Macros menu. Once built, they are inserted into the program in the Sequence menu.

Another way to build programs is to generate them on another system, then load them as ASCII files. For more information, refer to "Loading ASCII files" in chapter 6.

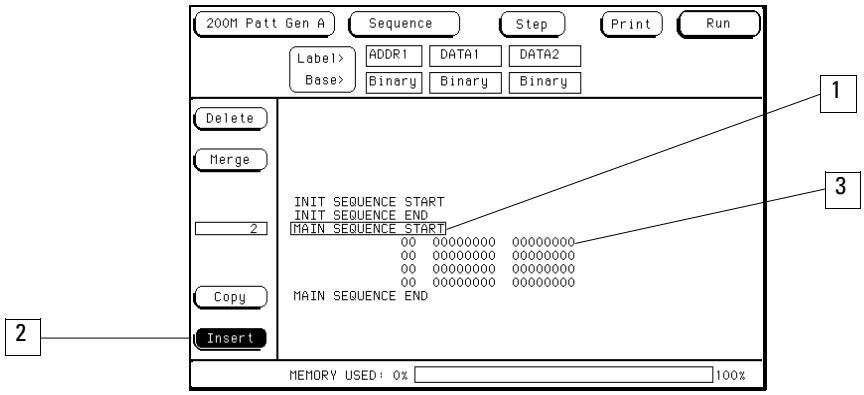
To build a main vector sequence

During a single run, the program vectors in the MAIN SEQUENCE are output to the system under test in an order of first vector to last vector. The data of the last vector is then held until run is selected again. During a repetitive run, the MAIN SEQUENCE loops until stop is selected.

1. From the Sequence menu, use the knob to highlight the first data row.
2. Select the Insert field once for each line of vectors you want inserted into the main sequence.

   These new lines of vectors provide fields to place data into for each label.
3. Select the data field, then type in a data value.

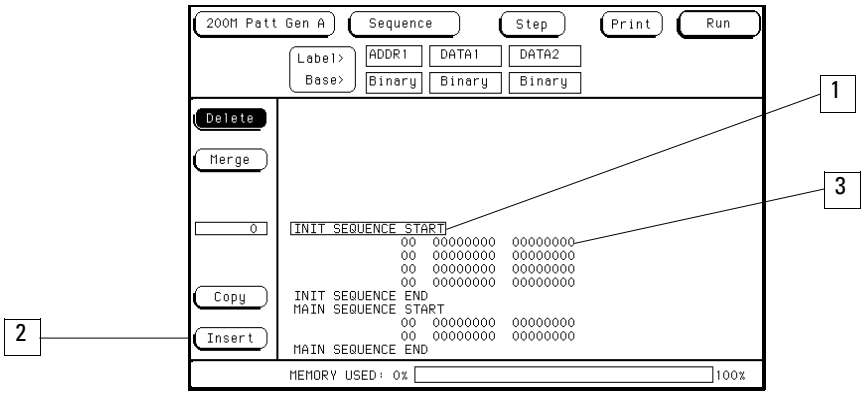
   If you will be adding data in many fields, use the Autoroll feature. Refer to "To use Autoroll" found later in this chapter.



To build an initialization sequence

Use the INIT SEQUENCE to place the system under test into a known initialization state. Default start and end program vectors are marked INIT SEQUENCE START and INIT SEQUENCE END. During a repetitive run, the initialization sequence is only executed the first time the program is run. The main sequence then loops repetitively.

1 From the Sequence menu, use the knob to highlight INIT SEQUENCE START.



2 Select the Insert field once for each new line of vectors you want inserted into the initialization sequence.

These new lines of vectors provide fields to place data into for each label.

3 Select the data field, then type in a data value.

If you are adding data in many fields, use the Autoroll feature. Refer to "To use Autoroll" found later in this chapter.

To edit a main or initialization sequence

1 Using the knob, highlight the vector you want to edit.

2 Select the data field you want to edit.

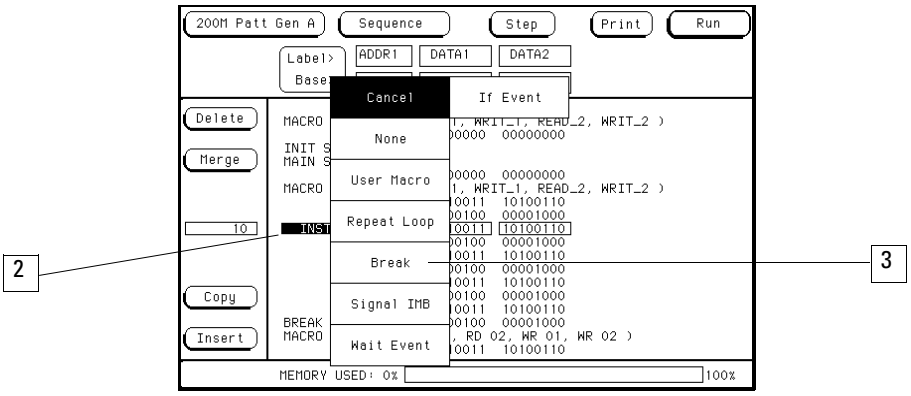
3 Select the new instruction or change the data value.

To include hardware instructions in a sequence

The following hardware instructions types are available:

- Break
- Signal IMB
- Wait Event
- If Event

1 Highlight the vector that you want output as a hardware instruction.
2 Select the INST field of the highlighted vector.
3 Select the desired hardware instruction type.
4 If required, select any qualifying actions for the hardware instruction.



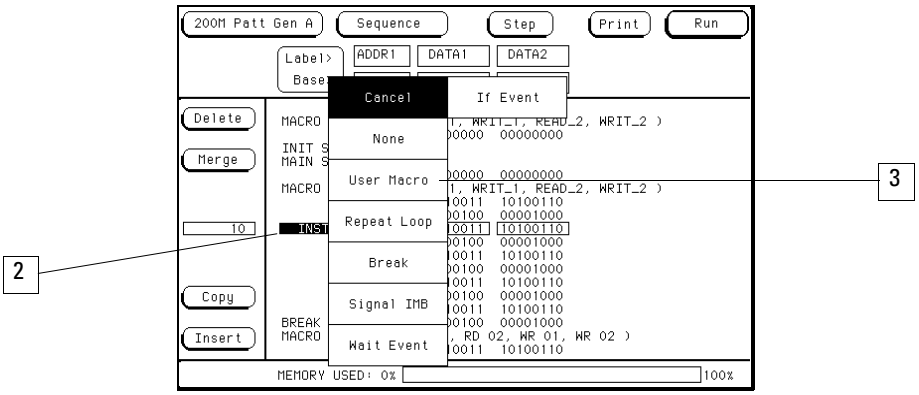
To include software instructions in a sequence

The following software instructions are available:

- User Macro
- Repeat Loop

If you are inserting a User Macro and have not yet built the macro, go to "To build a user macro" later in this chapter. Macros must be built before they can be inserted. To include these instructions in a sequence, use the following procedure.

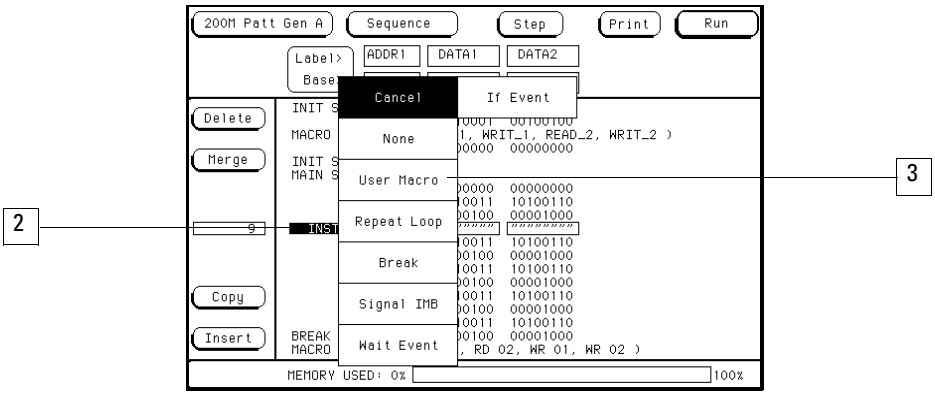
1. Highlight the vector that will be output at the time of the instruction.
2. Select the INST field.
3. Select the desired software instruction to insert.
4. If required, select any qualifying actions for the instruction.



To include a user macro in a sequence

If you have user macros, you can include them in the vector sequence using the following procedure. (If you have not yet built user macros, turn to the module "To build a user macro" to build needed macros.)

1. Insert a new vector where you want to place the user macro.
2. Highlight this new vector using the knob, then select the INST field.
3. Select the User Macro field.



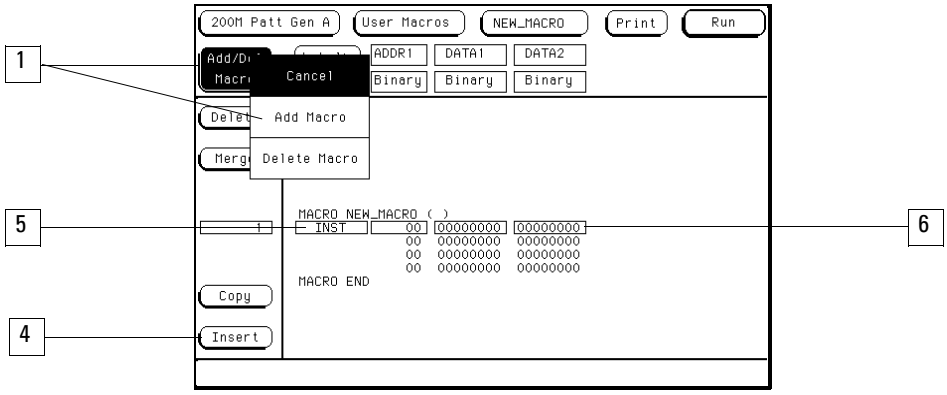
4. Select the user macro you want to insert from the list provided in the pop-up.

To build a user macro

Build macros for sequences of vectors you will want to use in multiple places. You can then insert these macros in INIT or MAIN sequences. Provide each macro with a name that will help you identify its function and make it easier to select from the list of macros you've built.

1. From the User Macros menu, select the Add/Del Macro field, then select ADD MACRO.
2. Select the MACRO field, then type the new macro name.
3. Add any desired parameters.

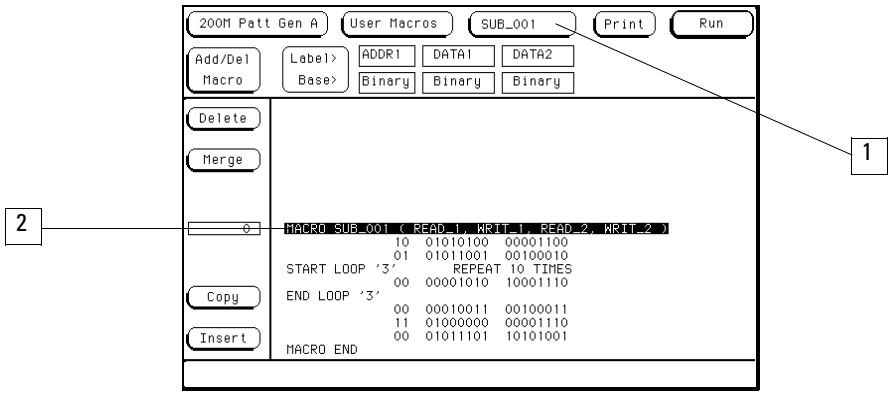
   Parameters are set when they are inserted into MAIN or INIT sequences. For more information on adding parameters, refer to "Adding parameters" found later in this chapter.
4. Select the Insert field to add as many vectors as needed into the macro.
5. Highlight the desired vector to modify.
6. Highlight a data field and insert the appropriate data.



To modify a macro name

If you rename a macro, the new macro name will be displayed in INIT and MAIN sequences where the macro has been used.

1. Select the macro to be renamed from the list of macros.
2. Highlight the first line of the macro, then select the field.
3. Modify the macro name, then select Done.



To edit a macro

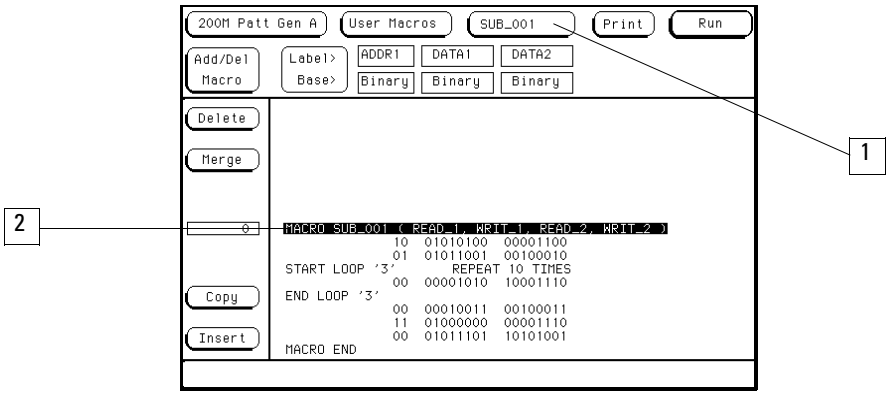
1. Highlight the vector you want to edit using the knob.
2. Select the field you want to edit.
3. Select the new instruction or change the data values using the pop-up.

To add, delete, or rename parameters

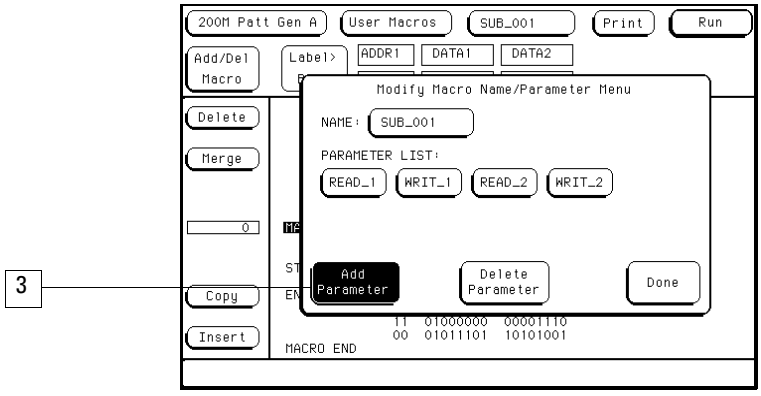
Parameters are set when they are inserted into MAIN or INIT sequences. The changes you make in the parameter list will appear every place in the INIT or MAIN sequences in which you have used that macro.

1 From the User Macros menu, select a macro from the list of macros.

2 Highlight the first line of the macro, then select the field.



3 To add a parameter, select Add Parameter, then select the new parameter field that appears and rename the parameter.

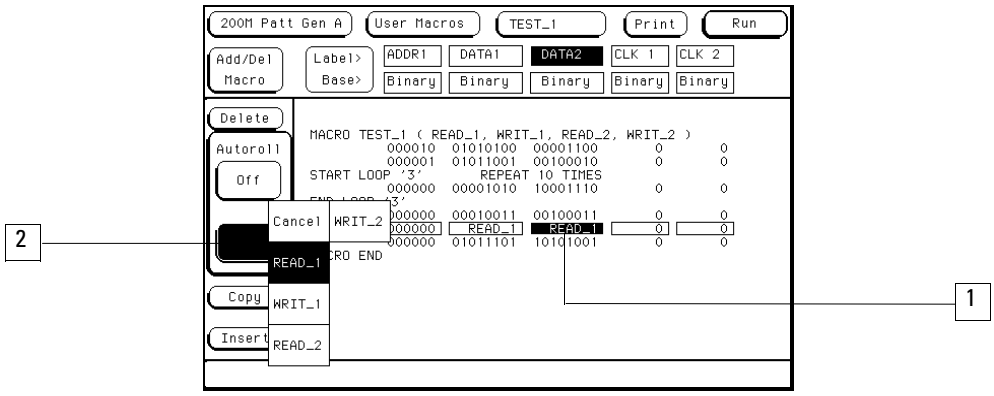


4 To delete a parameter from the parameter list, select a parameter name, then select Delete Parameter.

To place parameters in a vector

Once parameters are added to the parameter list, you insert them into data fields in macro vectors.

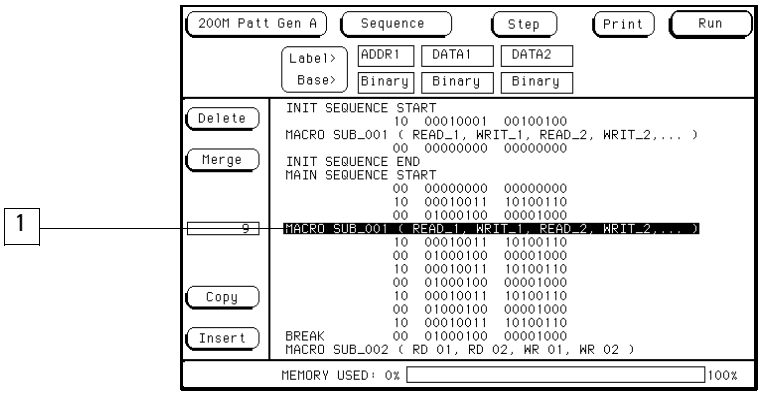
1. From the User Macro menu, select the desired data field in a vector.
2. Select the Set Param field. From the parameter list that appears, select the desired parameter to insert.



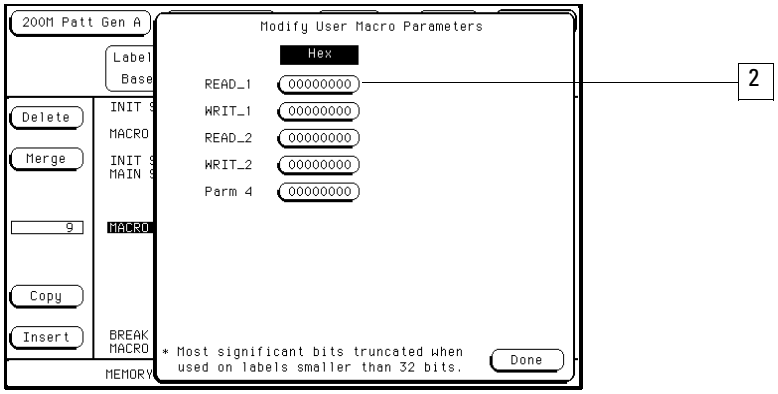
To enter or modify parameters

Each time you include a macro in an initialization or main sequence, you should enter the parameters for that particular instance. To enter or modify macro parameters, use the following procedure.

1 From the Sequence menu, highlight the line which contains the macro name, then select the field.



2 Enter or modify the parameter in the pop-up menu.



3 Select Done.

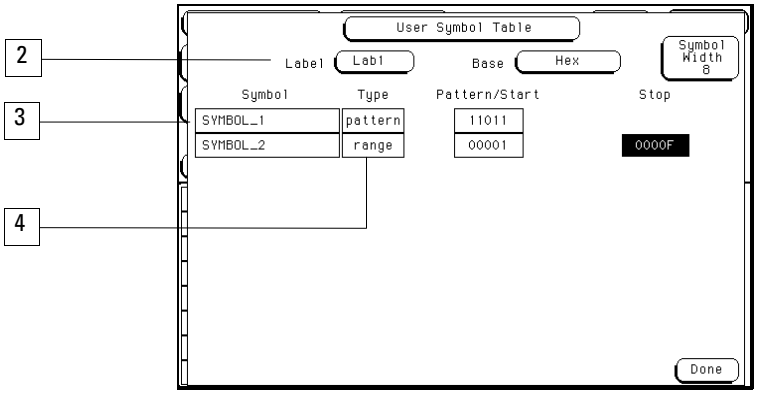
To build a User Symbol Table

You may want to build a symbol table to make inserting values into your program easier. You can name a symbol for one value in a label and insert that symbol into your vector sequence where you need it.

1. From the Format menu select the Symbols field at the right of the menu.
2. Set the desired Label, Base, and Symbol Width.

   Symbols are specific for a given label. Symbol width determines the width of the symbolic name displayed in the Sequence menu.
3. Select the Symbol field, then enter a name for the symbol.
4. Select the desired symbol Type, then enter the Pattern/Start and Stop values.

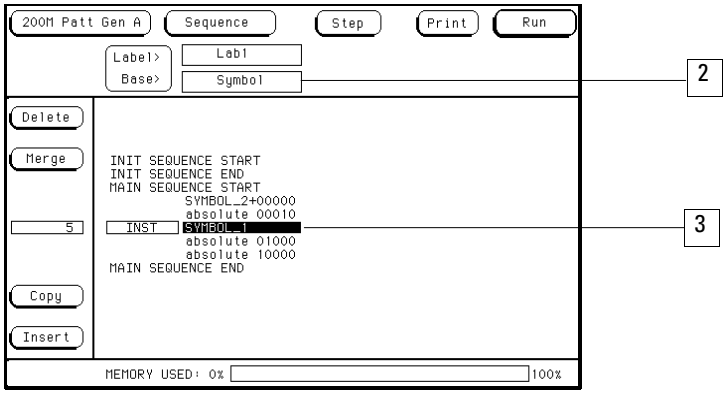
   The type is either a pattern or a range. A range provides a symbolic method for defining values within a specified range.
5. If you want to add more symbols, select the Symbol field, then select Add a Symbol, and repeat steps 2 through 4.
6. Select Done.



To include symbols in a sequence

Symbols must be created before they become available for insertion. See the task on the preceding page for more information.

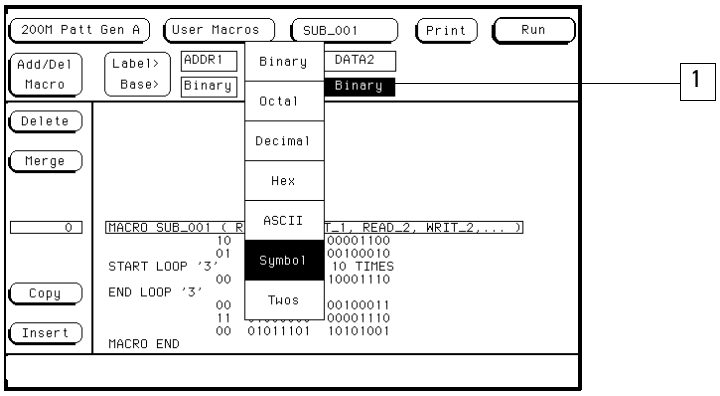
1. From the Sequence menu, select the Base field under the desired label where you want a symbol used.
2. From the Base selection list, select Symbol.
3. Highlight the desired vector, then select the data field.
4. Select the desired predefined symbol name.



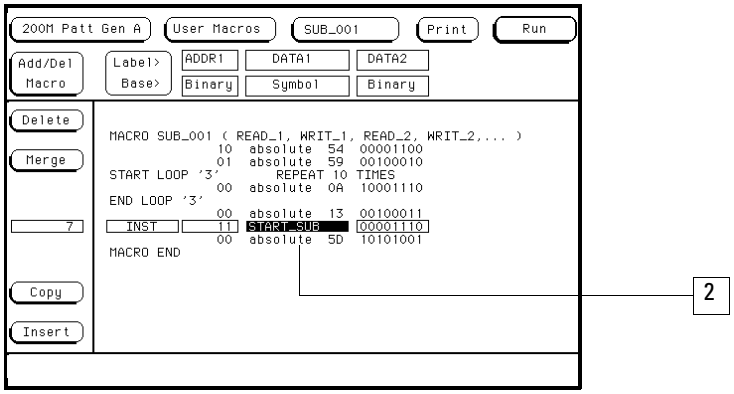
To include symbols in a macro

In the Format menu, you assign symbols to data under a given label. Once assigned, these symbols can be included under the same label in a macro.

1 From the User Macros menu, select the label Base field for any label you have preassigned symbols to. Then, select Symbol from the Base selection list.



2 Highlight any vector in the macro. Then, select the data field under the label that has the pre-assigned symbol.

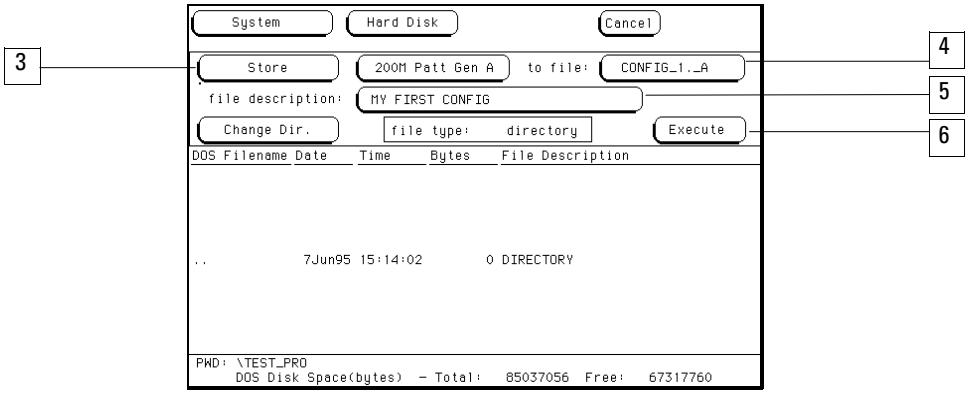


3 Select the symbol name you want displayed.

To store a configuration

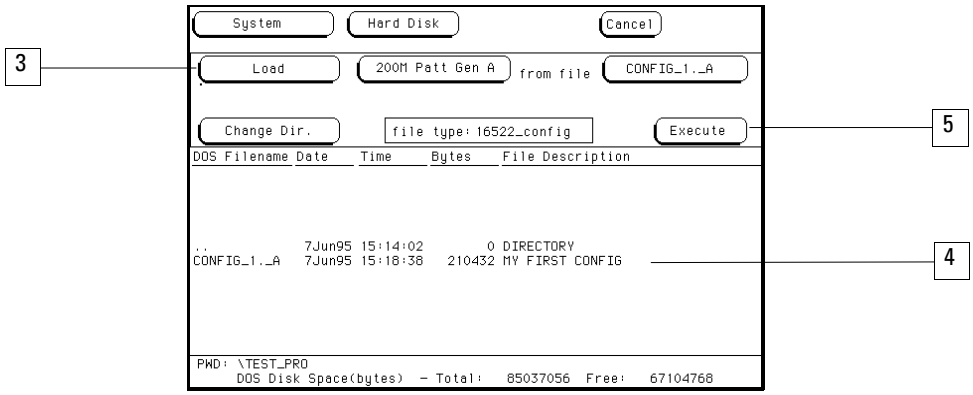
Once you have completed configuring the pattern generator, you can save that configuration to hard disk for future uses.

1. Change to the System module.
2. Select Configuration, then Hard Disk.
3. Select the Store operation, then 200M Patt Gen.
4. Select the **to file** field and type a name for the file.
5. Select the **file description** field and type in a description if desired.
6. Select Execute.



To load a configuration

1. Change to the System module.
2. Select Configuration, then Hard Disk.
3. Select the Load operation, then 200M Patt Gen.
4. Highlight the file to be loaded with the knob.
5. Select Execute.

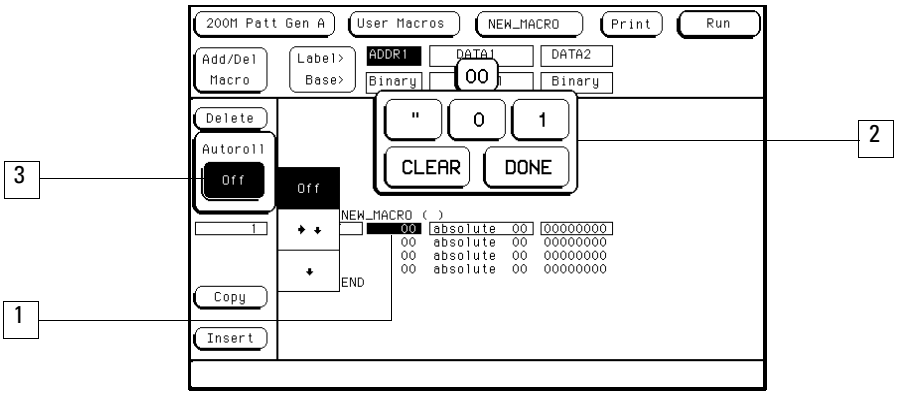


To use Autoroll

When Autoroll is used, each time you complete the process of adding data to a data field, the data entry focus changes to the next specified data field. The data entry keypad remains active, ready to define the next data field.

The following procedure shows you how to use Autoroll:

1. Select the first data field to define.
2. Enter the desired data using the pop-up keypad.
3. Select the Autoroll Off field, then select the desired roll direction.



4. As you continue to enter data and select Done, the data field focus rolls to the next data field automatically.

2

Agilent Technologies 16522A Reference

Agilent Technologies 16522A Reference

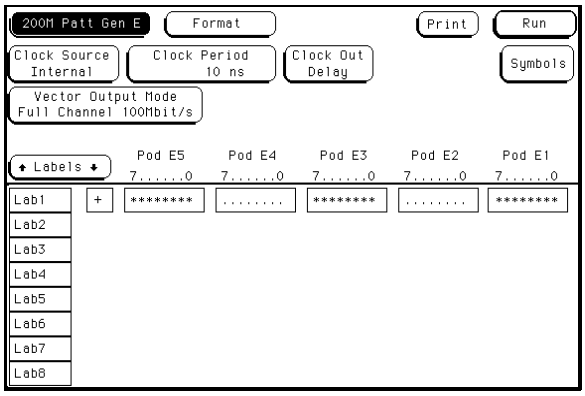
The Agilent Technologies 16522A is one in a family of modules that plugs into the Agilent Technologies 16500B/C Logic Analysis System. Many of the fields on the touch screen are identical to those on the other modules. Therefore, this section provides information only on those fields that are significantly different from the normal Agilent 16500B/C menu screens. This chapter provides reference information about the following menu screens.

- Format
- Sequence
- User Macros

For information about using the Agilent 16500B/C Logic Analysis System, see the *Agilent Technologies 16500/16501A Logic Analysis System User's Reference*.

The Format Menu

The Format menu allows you to configure the pattern generator with a clock source and parameters, generate a symbol table, select its output mode, assign which vector output channels are used, then group and label the vector output channels. The initial Format menu is shown below.



Clock Source

The Clock Source field toggles between internal and external. The internal clock source is supplied by the pattern generator and controls the frequency the vectors are output to the system under test.

The external clock is provided by you, or the system under test, and is input to the pattern generator through the CLK IN probe of a clock pod. By using an external clock, you synchronize the vector output of the pattern generator to the system under test.

No matter which clock is selected, vectors are output on the rising edge of the clock.

Clock Period (internal clock source)

This field toggles from Clock Period, when an internal clock source is selected, to Clock Frequency, when an external clock source is selected.

You select clock periods in steps of 1, 2, 2.5, 4, 5, 8, and 10. If the keypad is used to select a value between the step intervals, the value is rounded to the nearest interval.

The minimum clock period available with Vector Output Mode at Full Channel 100Mbit/s is 10 ns. The minimum clock period available with Vector Output Mode at Half Channel 200Mbit/s is 5 ns. Maximum clock period for either mode is 250 µs.

Clock Frequency (external clock source)

This field toggles from Clock Frequency, when an external clock source is selected, to Clock Period, when an internal clock source is selected. Set the clock frequency range to match the frequency of the external clock source.

If the Vector Output Mode is Full Channel 100Mbit/s, you are offered two clock frequency ranges:

- Less than 50 MHz
- Between 50 MHz and 100 MHz

If the Vector Output Mode is Half Channel 200Mbit/s, you are offered three clock frequency ranges:

- Less than 50 MHz
- Between 50 MHz and 100 MHz
- Greater than 100 MHz

If the external clock is faster than the frequency range selected, the Agilent 16522A will produce erroneous output vectors.

Clock Out Delay

The Clock Out Delay setting allows you to position the output clock with respect to data. The zero setting is uncalibrated and should be measured to determine the initial position with respect to the data. Each numerical change of one on the counter results in an approximate change of 1.3 ns.

Symbols

Touching the Symbols field brings up a pop-up menu that allows you to build a symbol table to use when putting data into the Sequence and User Macros menus. Providing symbol names to frequently used values allows you to enter the values more easily and to recognize these values by their symbol name rather than having to remember data values.

Vector Output Mode

The Vector Output Mode determines the channel width, available pods, and the frequency range for both the internal and external clock. The following table shows the difference between the Full Channel 100MBits/s mode and the Half Channel 200MBits/s mode.

	Full Channel 100 MBits/s	**Half Channel 200 MBits/s**
Pods Available	Pods 1, 2, 3, 4, 5	Pods 1, 3, 5
Maximum Channels	40; eight per pod	20; eight on pods 1, 3 and lower 4 on pod 5
Maximum External Clock Frequency	100 MHz	200 MHz
Maximum Internal Clock Frequency	100 MHz	200 MHz
Minimum External Clock Frequency	DC	DC
Minimum Internal Clock Frequency	4 kHz	4 kHz

Labels

Labels allow the user to group output channels from the data pods into a more logical configuration for creating vector data. The Agilent 16522A labels work in the same fashion as the labels for the logic analyzer products, with the exception that an output channel cannot be assigned to more than one label. For more information on how to use labels, refer to "To build a label" in chapter 1.

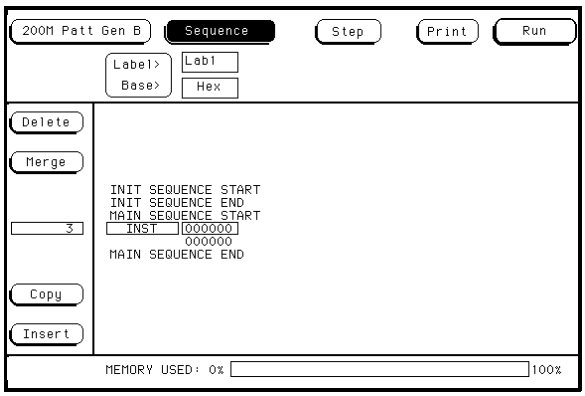
The Sequence Menu

Use the Sequence menu to build your test vector files. The Agilent 16522A provides for two sequences, an initialization sequence and a main sequence.

In single run mode the vectors are output from the first vector in the initialization sequence to the last vector of the main sequence. The last vector of the main sequence will be held at the outputs until run is executed again.

In repetitive run mode the vectors in the initialization sequence will be output from first to last one time, then the main sequence will repetitively output the vectors in that sequence until the stop field is pressed. The vector being output when stop is acknowledged by the module will be held at the outputs until run is executed again.

The initialization sequence can be empty in which case it will be ignored. The main sequence must contain at least two vectors to output. The Sequence menu is shown below.



INIT and MAIN Sequences

Use the knob to highlight individual lines in either vector sequences. When a line is highlighted, you can add data lines below it by selecting the Insert field. Selecting the INST field brings up a dialog box that lets you insert one of the instructions or user macros into the vector sequence.

An instruction is not allowed on the following vector lines of the sequence:

- The first vector of the INIT sequence.
- The first vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The vector prior to the IF block.
- The first vector of the IF block.
- The last vector of the IF block.
- The vector following the IF block.

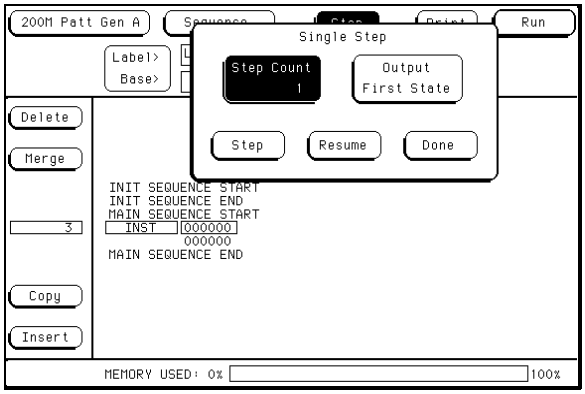
It should be noted that with the Vector Output Mode of Half Channel 200Mbit/s, the init sequence must contain a number of vectors that is divisible by four. If this is not the case, the first vector of the init sequence is duplicated to create the correct number of vectors.

With the Vector Output Mode of Full Channel 100 Mbit/s and vector frequencies greater than 50 MHz, the init sequence must contain an even number of vectors. Again if this is not the case, the first vector of the init sequence is duplicated to create the correct number of vectors.

Step

Use the Step field to step through your vector sequence to debug a critical set of vectors following a break instruction in the program sequence. Stepping will begin at the vector following the break instruction or the Output First State item can be pressed which will place the first vector of the sequence on the outputs. Stepping will then begin on the second vector of the sequence. When the last vector of the main sequence is encountered while stepping, the next step command places the first vector of the main sequence on the outputs.

When stepping through a sequence, breaks are ignored while valid branch and wait conditions are executed.



The Step Count field lets you set the number of steps to be executed, up to 100,000. The Output First State field reloads the hardware if necessary and places the first vector of the sequence on the outputs. The Resume field closes the Step pop-up menu and restarts the hardware without changing the previous run mode. Execution resumes from the point at which the sequence was stopped.

Delete

The Delete field lets you delete sequence lines. The From and To fields in the Delete pop-up menu lets you select line numbers with either the knob or another pop-up menu that appears when the From and To fields are touched twice.

When deleting vector rows, the INIT START, INIT END, MAIN START, and MAIN END cannot be deleted. Deleting all the vector rows from INIT START to MAIN END will reset the sequence to the power up state. The delete will not be performed if the results of the delete will place fewer than two vectors in the main sequence, or, if the delete will place an instruction on any of the following vectors:

- The first vector of the INIT sequence.
- The first vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The vector prior to the IF block.
- The first vector of the IF block.
- The last vector of the IF block.
- The vector following the IF block.

Merge

Touching the Merge field brings up a pop-up menu that lets you select sections of a previously created configuration file to merge after the current line in the sequence. The Input Drive field selects between the hard disk or the front floppy disk (if present). The Filename field displays a file browser pop-up menu that allows searching for the file to be used in the merge. The Merge Section field allows selection of the section of the configuration to be merged. Configuration sections are the main sequence and any macros that were created. The merge will not be performed in the following cases:

- Merge data exceeds the maximum number of sequence lines.
- Merge data requires more repeats than are available.
- Merge data requires more macro calls than are available.

Merge is not allowed in the following cases:

- Within a repeat loop.
- Within an IF block (starting with the vector prior to the if and ending with the vector following the IF).
- Between the start and first vector of the main sequence.
- After the last vector of the main sequence.
- Between the init and main sequence.
- Between the start and first vector of the init sequence.
- Within an empty init sequence (insert one vector in the init and merge after that vector).

Copy

Selecting the Copy field brings up a pop-up menu that lets you select vector sequence lines to be copied and a location to insert them after. The values in the Start, End, and Copy After fields can be selected with the knob or with the pop-up keypad that appears when the appropriate field is touched twice.

Copy will not be performed if you run out of macro calls (1000) or repeats (1000). Copy will not be allowed if the result of the copy places an instruction on one of the following vectors:

- The first vector of the INIT sequence.
- The first vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The vector prior to the IF block.
- The first vector of the IF block.
- The last vector of the IF block.
- The vector following the IF block.

Insert

Touching the Insert field adds another instruction line immediately below the line that is currently selected.

Instructions

User Macro The User Macro instruction brings up a list of current user macros you can insert. Macros are inserted at the current line and expanded at run time.

If the macro selected has parameters, a second pop-up menu will be displayed to allow setting of the passed parameter values. Parameters are passed as 32 bit values and the most significant bits are truncated when the data is applied to labels with less than 32 bits.

Once inserted, the passed parameters of a macro may be altered by selecting that macro again and changing the data. A macro can only be removed from the sequence by using the delete field.

Repeat Loop The Repeat Loop instruction inserts the start and end of a repeat loop using the current vector row as the data. Once the loop has been created, vectors can be inserted or copied into the loop to change the size. A repeat loop will be expanded at run time into individual vectors. The number of repetitions of the loop (maximum 20000) is set when the repeat is first inserted into the sequence and can be altered by selecting the start of the repeat to get the Repeat Count pop-up menu. Both the start and end of a repeat loop will be removed from the sequence if either is included in the delete block.

Break The Break instruction causes a break at the current vector. In single run mode, this instruction halts the sequence and holds the outputs at the break vector's value. In repetitive run mode, this instruction pauses the sequence at the current vector momentarily, then continues. The duration of the pause is dependent on the activity of other modules in the frame.

Signal IMB The Signal IMB instruction creates a signal for the IMB bus of the 16500B/C at the current vector, allowing the Agilent 16522A to trigger other modules in the frame. Multiple signal IMB instructions may be placed in the sequence, but only the first signal IMB will be executed.

Wait Event The Wait Event instruction halts the execution of the program sequence until the event is received by the hardware. Selecting this instruction brings up a pop-up menu that allows you to set four data patterns and select one of them or an IMB signal as the event the pattern generator is waiting for.

An external wait event is the ORing of the three input lines on the clock pod. The first wait IMB is the only one recognized since IMB does not allow pulsing.

If Event The If Event instruction is only available in Full Channel 100 Mbit/s mode with clock frequencies of 50 MHz and lower. Only one If Event is allowed in a sequence program. If a new if instruction is placed in a program sequence, a message will appear stating that only one if instruction is allowed. To add a new if instruction, the old one must be deleted.

The If event uses either the IMB or the same external clock pod input lines as the Wait event. If the condition is true at the If event, then the data in the If block is output, otherwise it is skipped.

The If event takes the current data line and duplicates as in the following example:

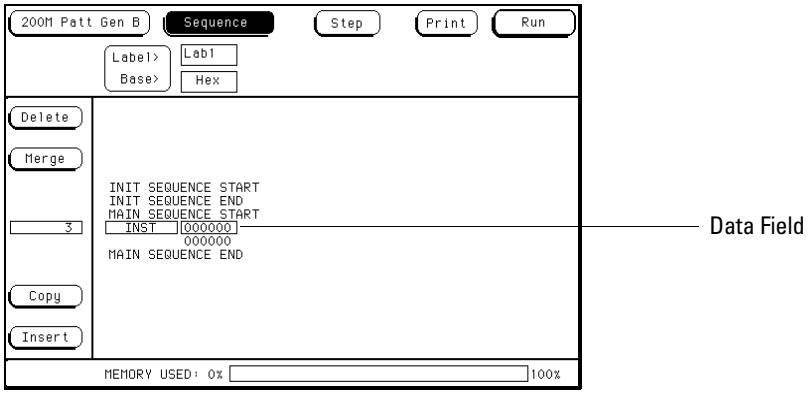
```
current data
        IF
          current data
          current data
        END IF
current data
```

These vectors are now restricted. They cannot have instructions. Delete and Copy operations that result in instructions being placed on these vectors will not be allowed. The If can be removed by deleting either the start or end of the If block.

It should be noted that when an If statement is in the sequence, repetitive runs of the sequence have latency between the last and the first vectors of the main sequence. This latency is dependent on the activity of other modules in the frame, thus varying from run to run.

Data Field

Selecting the data field to the right of the instruction field brings up a pop-up menu that allows you to insert vector data. ASCII-based data cannot be edited, and ASCII- and Symbols-based data cannot be autorolled.



Autoroll

The Autoroll field is provided to reduce the number of keystrokes required to enter data into the sequence or a macro. When a data field is edited (except in the last vector row), the Autoroll field appears to the left of the data entry pop-up menu. Autoroll can move from left to right across the labels on the display with an automatic line feed, or it can move down a label from vector row to row. When the last vector row of the sequence is encountered, Autoroll will stop the editing process, or the Autoroll can be halted at any point in the editing process by turning the autoroll function off.

For more information on how to use the Autoroll, refer to "To use Autoroll" in chapter 1.

MEMORY USED

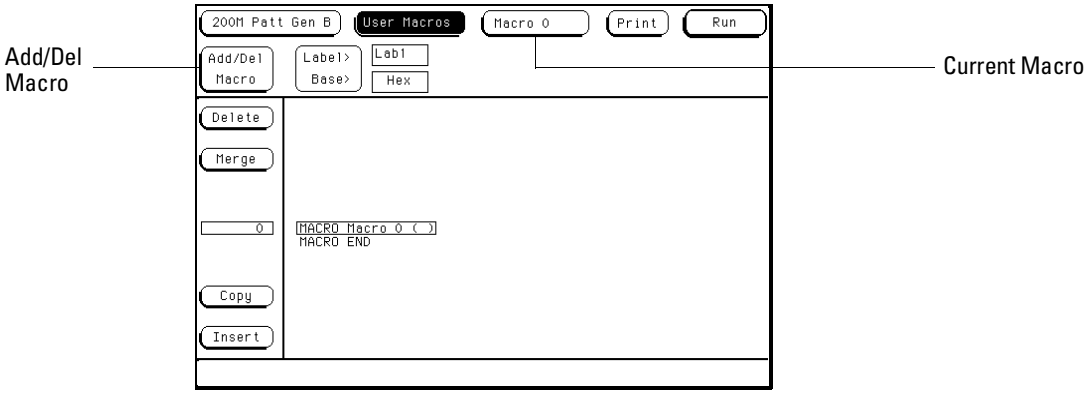
The Memory Used bar shows in percent how much memory is being used by the vector sequence entered. It should be noted that Repeat Loops and User Macros may suddenly increase the amount of memory used because they are expanded at run time. It is critical to use the Memory Used bar to obtain an idea of how much memory will be needed at run time.

The User Macros Menu

The User Macros menu is used to create new macros and edit existing macros. Macro 0 is the default macro and always exists. Macros enable you to define a pattern sequence once, then insert the macro by name wherever it is needed.

Macros can also have parameters passed to them. Parameters let you create a generic macro. For each instance of a macro, you specify unique values for the parameterized variable. Each macro can have up to 10 parameters. Up to 100 different macros can be defined for use in a single stimulus program.

Differences between User Macros and the INIT and MAIN sequences are that macros cannot use the If instruction and a macro cannot call another macro.



This section only covers the two unique fields in the User Macros menu and using parameters. For information about other fields in this menu, refer to the discussion of these fields in the previous section on the Sequence Menu.

Macro 0 (current macro field)

Touching this field brings up a list of macros that have been created and are available to insert into the Sequence menu. If you want to edit a previously built macro or to view it, select that macro from the list and it will appear in the main part of the display.

Add/Delete Macro

Selecting this field brings up a pop-up menu that allows you to choose between adding a new macro or deleting one. If you select Add Macro, a new blank macro appears in the display with a new Macro "n" name.

To rename the new macro, select the macro name field, and in the pop-up menu that appears, type in a new name. You can also add parameters to the new macro in the same pop-up menu. For more information on parameters, refer to "Using Parameters" below.

If you select Delete Macro, you will be presented with a list of current macros. Select the macro you want to delete from the list. If the macro you delete is being used in the MAIN sequence, it will be removed from the sequence. If you try to delete the first macro, it will be removed from the MAIN sequence but not from the macro list.

Using Parameters

Parameters are used to pass values into macros. A major benefit in passing parameters is that you keep a macro's functionality generic and still direct specific action identified by parameters. Think of parameters as the only part of a macro that changes as the macro is reused.

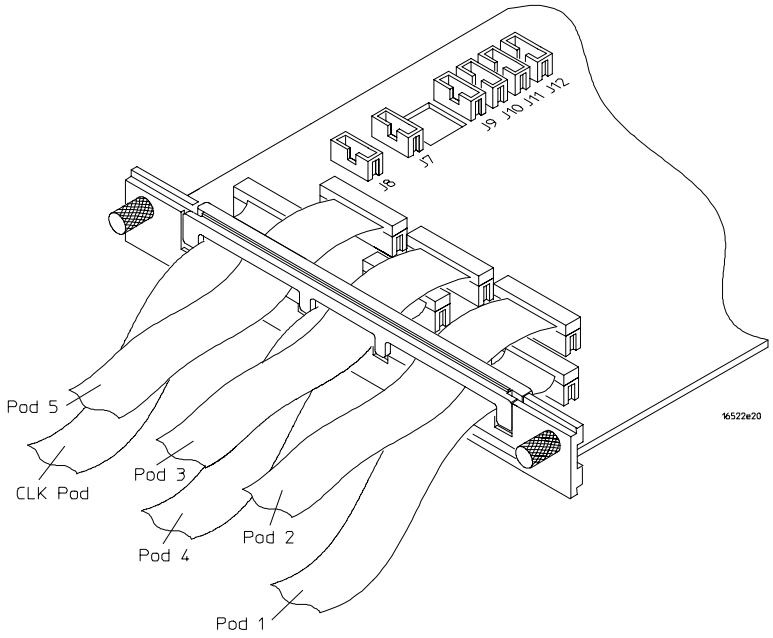
You create the macro and add parameters to the macro in the User Macro menu. You then insert the macro and assign parameter values in the Sequence menu. As you reuse the macro in other places in the sequence, simply change parameter values to the new specific values.

Macro parameters are passed as 32 bit values. If fewer than 32 bits are assigned to a label in the Format menu, the most significant bits of the parameter value are truncated when the parameter is used as data for the given label.

For more information on using macros and parameters, refer to chapter 1.

Pod Numbering

The Agilent 16522A pods are numbered as shown in the figure below.

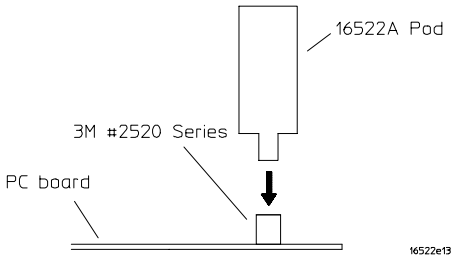


Connecting Pods Directly to a PC Board

To connect the pattern generator pods directly to the PC board, use one of the following two methods. Both methods require a 3M 2520-series, or similar alternative connector be installed on the PC board.

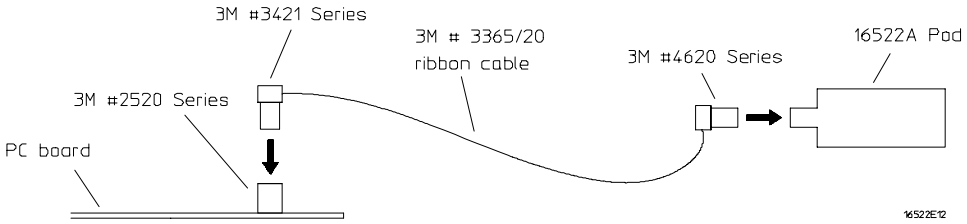
Direct pod to board connection

Simply plug the pod directly into the 3M 2520-series, or similar alternative connector on the PC board.



Jumper cable to pod connection

Use this method when you have clearance problems on the PC board. Construct a flat ribbon cable and connect as shown below.



Equivalent connectors can be obtained from sources other than 3M.

Output Pod Characteristics

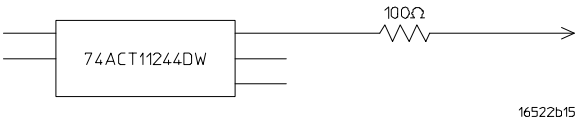
The following equivalent circuit information is provided to help you select the appropriate clock and data pods for your application.

Agilent 10461A TTL Data Pod

Output type	10H125 with 100 ohm in series
Maximum clock	200 MHz
Skew	Typical <2 ns; worst case 4 ns (note 1)
Recommended lead set	Agilent 10474A

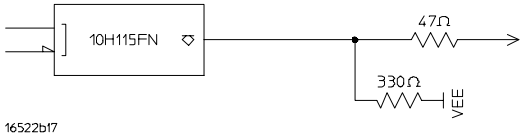
Agilent 10462A 3-State TTL/CMOS Data Pod

Output type	74ACT11244 with 100 ohm in series 10H125 on non 3-state channel 7 (note 2),
3-state enable	negative true, 100K ohm to GND, enabled on no connect
Maximum clock	100 MHz
Skew	Typical <4 ns; worst case 12 ns (note 1)
Recommended lead set	Agilent 10474A



Agilent 10464A ECL Data Pod (terminated)

Output type	10H115 with 330 ohm pulldown, 47 ohm in series
Maximum clock	200 MHz
Skew	Typical <1 ns; worst case 2 ns (note 1)
Recommended lead set	Agilent 10474A

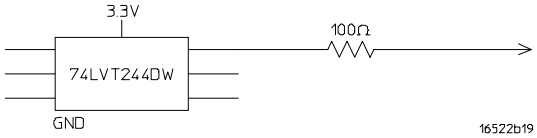


Agilent 10465A ECL Data Pod (unterminated)

Output type	10H115 (no termination)
Maximum clock	200 MHz
Skew	Typical <1 ns; worst case 2 ns (note 1)
Recommended lead set	Agilent 10347A

Agilent 10466A 3-State TTL/3.3 Volt Data Pod

Output type	74LVT244 with 100 ohm in series 10H125 on non 3-state channel 7 (note 2)
3-state enable	negative true, 100K ohm to GND, enabled on no connect
Maximum clock	200 MHz
Skew	Typical <3 ns; worst case 7 ns (note 1)
Recommended lead set	Agilent 10474A

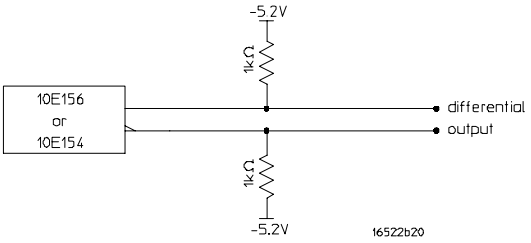


Note 1 Typical skew measurements made at pod connector with approximately 10 pF/50K ohm load to GND; worst case skew numbers are a calculation of worst case conditions through circuits. Both numbers apply to any channel within a single-card or multiple-card module.

Note 2 Channel 7 on the 3-state pods has been brought out in parallel as a non 3-state signal. By looping this output back into the 3-state enable line, the channel can be used as a 3-state enable.

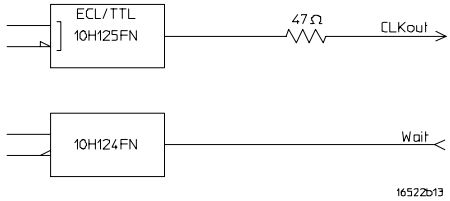
Data Cable Characteristics Without a Data Pod

The Agilent 16522A data cables without a data pod provide an ECL-terminated
(1 KΩ to -5.2 V) differential signal. These are usable when received by a differential receiver, preferably with a 100 ohm termination across the lines. These signals should not be used single ended due to the slow fall time and shifted voltage threshold (they are not ECL compatible).



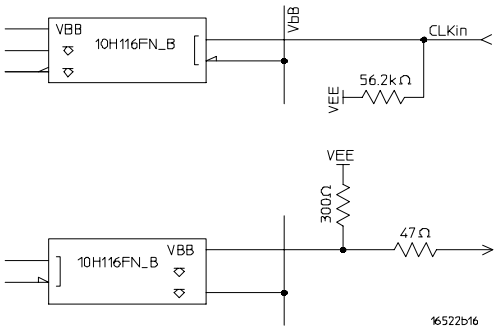
Agilent 10460A TTL Clock Pod

Clock output type	10H125 with 47 ohm series; true & inverted
Clock output rate	100 MHz maximum
Clock out delay	11 ns maximum in 9 steps
Clock input type	TTL - 10H124
Clock input rate	DC to 100 MHz
Pattern input type	TTL - 10H124 (no connect is logic 1)
Clk-in to clk-out	Approx. 30 ns
Patt-in to recognition	Approx. 15 ns + 1 clk period
Recommended lead set	Agilent 10474A



Agilent 10463A ECL Clock Pod

Clock output type	10H116 differential unterminated; and differential with 330 ohm to -5.2v and 47 ohm series
Clock output rate	200 MHz maximum
Clock out delay	11 ns maximum in 9 steps
Clock input type	ECL - 10H116 with 50 KΩ to -5.2 V
Clock input rate	DC to 200 MHz
Pattern input type	ECL - 10H116 with 50 KΩ (no connect is logic 0)
Clk-in to clk-out	Approx. 30 ns
Patt-in to recognition	Approx. 15 ns + 1 clk period
Recommended lead set	Agilent 10474A



3

Programming the Agilent 16522A

Programming the Agilent 16522A

This chapter, combined with the *Agilent 16500 Programmer's Guide* manual, provides you with the information needed to program the
Agilent 16522A pattern generator module. Each module has its own manual to supplement the mainframe manual since not all mainframes will be configured with the same modules.

This chapter is organized into seven sections. The first section discusses:

- Programming overview and instructions to help you get started
- Mainframe system commands that are frequently used with the pattern generator module
- Agilent 16522A Pattern Generator command tree
- Alphabetic command-to-subsystem directory

The next section contains the module level commands and the following four sections contain the subsystem commands for the pattern generator. The final section contains information on the SYSTem:DATA and SYSTem:SETup commands for this module.

Error messages for the Agilent 16522A are included in generic system error messages and are in the *Agilent 16500 Programmer's Guide*.

Programming commands are derived from front panel operation. These operations are described in detail in chapter 2. You should be familiar with the front panel operations before attempting to program the pattern generator.

Programming Overview

This section introduces you to the basic command structure used to program the pattern generator.

Example Pattern Generator Program

A typical pattern generator program includes the following tasks:

- select the appropriate module
- set program parameters
- define a pattern generator program
- run the pattern generator program

The following example program generates a pattern using two of the output pods of the master card:

```
10 OUTPUT XXX;":SELECT 1"
20 OUTPUT XXX;":FORMAT:REMOVE ALL"
30 OUTPUT XXX;":FORMAT:LABEL 'A',POSITIVE,127,0"
40 OUTPUT XXX;":FORMAT:LABEL 'B',POSITIVE,0,255"
50 OUTPUT XXX;":SEQ:REMOVE ALL"
60 OUTPUT XXX;":SEQ:INSERT 0,NOOP,'#H7F','#HFF'"
70 OUTPUT XXX;":SEQ:INSERT 4,NOOP,'#H7F','#HFF'"
80 OUTPUT XXX;":RMODE REPETITIVE"
90 OUTPUT XXX;":START"
100 END
```

The three Xs (XXX) after the OUTPUT statement in the above example refer to the device address required for programming over either GPIB or RS-232-C. Refer to your controller manual and programming language reference manual for information on initializing the interface.

Program Comments

Line 10 selects the pattern generator in slot A

Line 20 removes all labels previously assigned

Line 30 assigns label 'A', positive polarity and assigns the seven least significant bits of pod A5

Line 40 assigns label 'B' and assigns all eight bits of pod A4

Line 50 removes all program lines

Line 60 inserts a new line (after line 0) in the INIT SEQUENCE portion of the program.

Line 70 inserts a new line (after line 4) in the MAIN SEQUENCE portion of the program. Recall that the default MAIN SEQUENCE already has two lines of program

Line 80 Sets the RMODE to repetitive. If the program is to be run only once, select the :RMODE SINGLE command.

Line 90 Starts the program.

Selecting the Module

Before you can program the pattern generator, you must first "select" it, otherwise, there is no way to direct your commands to the pattern generator.

To select the module, use the system command :SELect , followed by the numeric reference for the slot location of the pattern generator(1...10 refers to slot A...J respectively). For example, if the pattern generator master card is in slot E, then the command:

```
:SELect  5
```

would select this module. For more information on the SELect command, refer to the *Agilent 16500 Programmer's Guide*.

Mainframe Commands

These commands are part of the Agilent 16500B/C mainframe system and are mentioned here only for reference. For more information on these commands, refer to the *Agilent 16500 Programmer's Guide*.

Query

`CARDcage?`

The CARDcage query returns a string which identifies the modules that are installed in the mainframe. The returned string is in two parts. The first five two-digit numbers identify the card type. The identification number for the pattern generator master is 25, while the pattern generator expander uses 24. A "-1" in the first part of the string indicates no card is installed in the slot.

The five single-digit numbers in the second part of the string indicate in which slots cards are installed and where the master card is located.

Example

13,13,-1,-1,25,1,1,0,0,5

A returned string of 13,13,-1,-1,25,1,1,0,0,5 means that an oscilloscope is loaded in slot A and slot B. The next two slots (C and D) are empty (-1). Slot E contains a pattern generator module (ID number 25).

The next group of numbers (1,1,0,0,5) indicate that a two card module is installed in slots A and B with the master card in slot A. The "0" indicates an empty slot or the module software is not recognized or not loaded. The last digit (5) in this group indicates a single module card is loaded in slot E. Complete information for the CARDcage query is in the *Agilent 16500 Programmer's Guide*.

Subsystem `INTermodule Subsystem`

The INTermodule Subsystem commands are used to specify intermodule arming between multiple modules.

Subsystem `MMEMory`

The MMEMory Subsystem provides access to both disk drives for loading and storing configurations.

Command/Query `MENU`

The MENU command selects a new displayed menu. The first parameter (X) specifies the desired module. The optional second parameter specifies the desired menu in the module (defaults to 0 if not specified). The query returns the currently selected (and displayed) menu.

For the Agilent 16522A Pattern generator:

- X,0 - Format Menu
- X,1 - Sequence Menu
- X,2 - Macro Menu

X = slot number that contains the pattern generator master card.

Command/Query `RMODe`

The RMODe command specifies the run mode (single or repetitive) for a module or intermodule. If the selected module is configured for intermodule, the intermodule run mode will be set by this command. The RMODe query returns the current setting.

Command/Query `SELect`

The SELect command selects which module or intermodule will have parser control. SELect 0 selects the intermodule, SELect 1 through 10 selects modules A through J respectively. Parameters -1 and -2 select software options 1 and 2. The SELect query returns the currently selected module.

Command `STARt`

The STARt command starts the specified module or intermodule. If the specified module is configured for intermodule, STARt will start all modules configured for intermodule.

Command `STOP`

The STOP command stops the specified module or intermodule. If the specified module is configured for intermodule, STOP will stop all modules configured for intermodule.

Query `SYSTem:ERRor?`

The SYSTem:ERRor query returns the oldest error in the error queue. In order to return all the errors in the error queue, a simple FOR/NEXT loop can be written to query the queue until all errors are returned. Once all errors are returned, the queue will return zero.

Command/Query `SYSTem:PRINt`

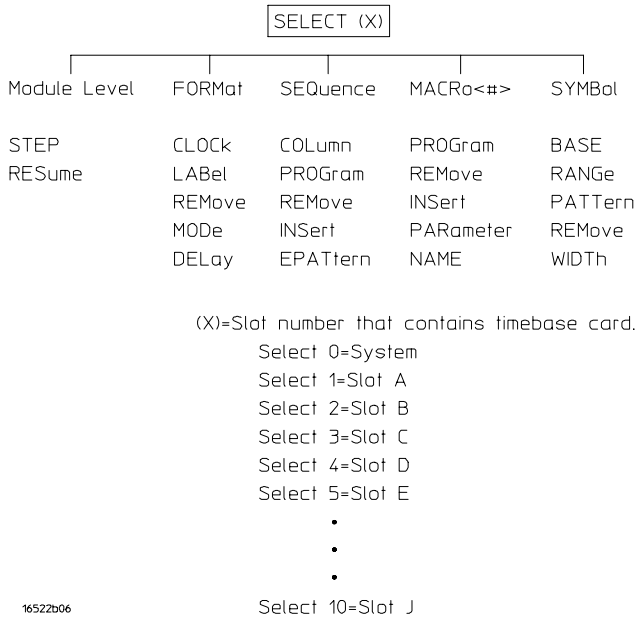
The SYSTem:PRINt command initiates a print of the screen or listing buffer over the current printer communication interface. The SYSTem:PRINt query sends the screen or listing buffer data over the current controller communication interface.

Command Set Organization

The command set for the Agilent 16522A is divided into four separate subsystems. The subsystems are: FORMat, SEQuence, MACRo, and the SYMBol subsystem. Each of the subsystems commands are covered in their individual sections later in this chapter.

Each of these sections contain a description of the subsystem, syntax diagrams and the commands in alphabetical order. The commands are shown in long form and short form using upper and lower-case letters. For example, FORMat indicates that the long form of the command is FORMAT and the short form is FORM. Each of the commands contain a description of the command and its arguments, the command syntax, and a programming example.

The following figure shows the command tree for the pattern generator module.



Pattern Generator Command Tree

Table 3-1 shows the alphabetical command to subsystem directory.

Table 3-1 Alphabetical Command to Subsystem Directory

Command	Where Used
BASE	SYMBol
CLOCk	FORMat
COLumn	SEQuence
DELay	FORMat
EPATtern	SEQuence
INSert	MACRo, SEQuence
LABel	FORMat
MODe	FORMat
NAME	MACRo
PARameter	MACRo
PATTern	SYMBol
PROGram	SEQuence, MACRo
RANGe	SYMBol
REMove	FORMat, SEQuence, MACRo, SYMBol
RESume	Module Level
STEP	Module Level
WIDTh	SYMBol

Module Status Reporting

Each module reports its status to the Module Event Status Register (MESR<N>) which in turn reports to the Combined Event Status Register (CESR) in the Agilent 16500B/C mainframe (see *Agilent 16500 Programmer's Guide*). The Module Event Status Register is enabled by the Module Event Status Enable Register (MESE<N>).

The following descriptions of the MESE<N> and MESR<N> commands provide the module specific information needed to enable and interpret the contents of the registers

MESE<N>

The MESE<N> command sets the Module Event Status Enable register bits. The MESE register contains a mask value for the bits enabled in the MESR register. A one in the MESE will enable the corresponding bit in the MESR register; a zero will disable the bit.

The first parameter after the command specifies the module (<N> = 1...10 refers to the module in slot A...J). The second parameter specifies the enable value.

The MESE query returns the current setting.

Refer to table 3-2 Module Event Status Register for bits, bit weights, and what each bit masks in the module.

Command Syntax: `:MESE<N><enable_mask>`

`<N>` `{1|2|3|4|5|6|7|8|9|10}` number of slot in which the module resides

`<enable_mask>` integer 0 to 255

Example

```
OUTPUT XXX;":MESE5 2"
```

This example enables bit one for slot E.

Query Syntax: `:MESE<N>?`

Returned Format: `[MESE] <enable_mask><NL>`

Example

```
10 OUTPUT XXX;":MESE2?"
20 ENTER XXX; Mes
30 PRINT Mes
40 END
```

This example reads status mask for slot B.

Table 3-2 Module Event Status Enable Register

Bit	Weight	Enables
7	128	Not Used
6	64	Not Used
5	32	Not Used
4	16	Not Used
3	8	Not Used
2	4	Not Used
1	2	Not Used
0	1	Run Complete

The Module Event Status Enable Register contains a mask value for the bits to be enabled in the Module Event Status Register (MESR). A one in the MESE enables the corresponding bit in the MESR, a zero disables the bit.

MESR<N>

The MESR<N> query returns the contents of the Module Event Status register.

Reading the register clears the Module Event Status Register.

Table 3-3 shows each bit in the Module Event Status Register and its bit weight for this module. When you read the MESR, the value returned is the total bit weights of all bits that are high at the time the register is read.

The parameter 1...10 refers to the module in slot A...J respectively.

Query Syntax: `:MESR<N>?`

Returned Format: `[MESR]<status><NL>`

`<N>` `{1|2|3|4|5|6|7|8|9|10}` number of slot in which the module resides

`<status>` 0 to 255

Example

```
10 OUTPUT XXX;":MESR2?"
20 ENTER XXX; Mer
30 PRINT Mer
40 END
```

This example reads status register for slot B

Table 3-3 **Module Event Status Register**

Bit	Weight	Condition
7	128	Not Used
6	64	Not Used
5	32	Not Used
4	16	Not Used
3	8	Not Used
2	4	Not Used
1	2	Not Used
0	1	1 = Run complete 0 = Run not complete

The MESR bit will be set at the end of the program or if a BREAK instruction is encountered within the program.

The Module Level Commands control the operation of pattern generator programs. The two Module Level Commands are STEP and RESume.

Module Level Syntax Diagram

count = *integer from 1 to 100,000 specifying the number of vectors stepped.*

STEP

Command/Query

The STEP command consists of four types: the STEP command, the STEP Count command, the STEP query, and the STEP FSTate command.

The STEP command causes the pattern generator to step through the number of vectors specified by the STEP Count command. If one of the instructions is BREAK, STEP will not stop for it.

The STEP Count command specifies the vector range for the STEP command. The valid vector range for the STEP Count command is from 1 to 100,000. The default is 1. If <count> is greater than the number of lines in the program, STEP will loop back to the beginning until it has stepped through <count> number of vectors.

The STEP query returns the current count.

The STEP FSTate (step first state) command outputs the first vector of the sequence.

If the vectors have been changed since last run, they must be loaded into the hardware with either the :START command or :STEP FSTate.

STEP command Syntax

```
:STEP
```

Example

```
OUTPUT XXX;":STEP"
```

STEP Count command Syntax

```
STEP <count>
```

<count> an integer from 1 to 100,000 specifying the number of vectors stepped.

Example

```
10 OUTPUT XXX;":STEP 20"
20 OUTPUT XXX;":STEP"
```

This example sets the step count to 20 in line 10, then in line 20 begins the step command through the number of lines specified in line 10.

Query `:STEP?`

Returned Format `[STEP] <count>`

Example

```
10 DIM Sc$[100]
20 OUTPUT XXX;":STEP?"
30 ENTER XXX;Sc$
40 PRINT Sc$
50 END
```

This example queries and prints the step count.

STEP FSTate command Syntax `:STEP FSTate`

Example

```
OUTPUT XXX;":STEP FSTATE"
```

RESume

Command

When the pattern generator encounters a BREAK instruction, program execution is halted. The RESume command allows the program to continue until another BREAK instruction is encountered, or until the end of the program is reached.

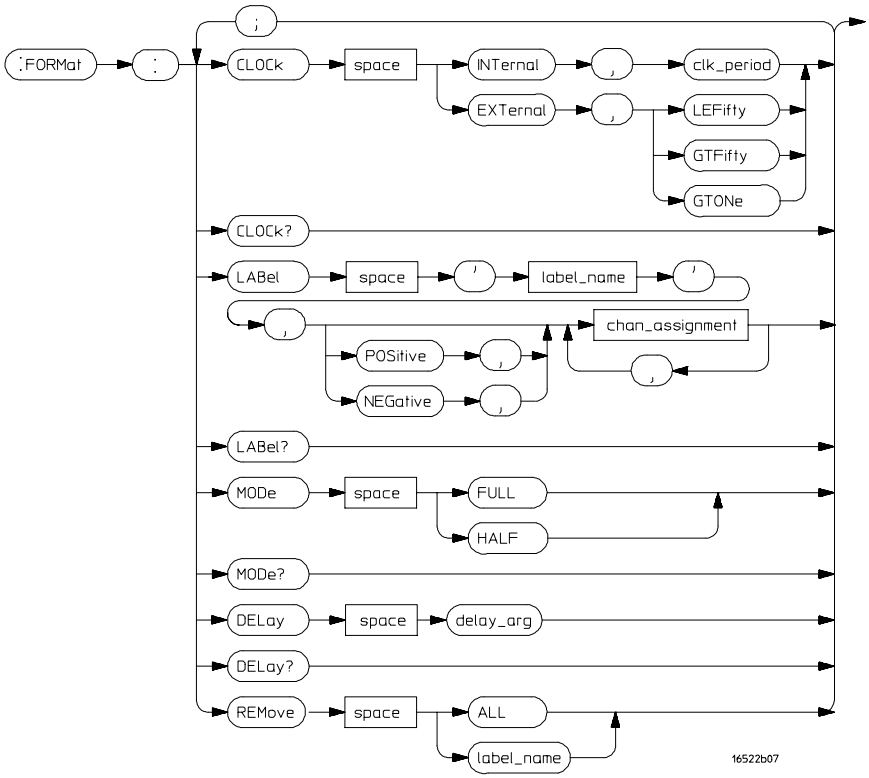
Command Syntax

```
:RESume
```

Example

```
OUTPUT XXX;":RESUME"
```

The commands of the Format subsystem control the pattern generator values such as data output rate, delay, and the channels that you want to be active. The Format subsystem also lets you specify the clock source and allows you to group channels together under a common, user-defined name.



Format Subsystem Syntax Diagram

label name = *a string of up to 6 alphanumeric characters*
chan_assignment = *an integer from 0 to 255*
clk_period = *a real number specifying the internal clock period*
delay_arg = *a integer specifying the delay*

CLOCk

Command/Query

The CLOCk command is used to specify the clock source for the pattern generator. The choices are INTernal or EXTernal. With an internal clock source, the clock period must also be specified (real number value).

With an external clock source, the clock frequency range must be specified as one of the following:

- Less than or equal to 50 MHz (LEFifty)
- Greater than 50 MHz and less than or equal to 100 MHz (GTFifty)
- Greater than 100 MHz (GTONe)

The maximum clock rate is limited by the output channel mode selected (see FORMat:MODe command).

Command Syntax

```
:FORMat:CLOCk INTernal,<clk_period>
:FORMat:CLOCk EXTernal,{LEFifty|GTFifty|GTONe}
```

`<clk_period>` a real number clock period that corresponds to the front-panel selectable clock period values.

Query Syntax

```
:FORMat:CLOCk?
```

Returned Format

```
[:FORMat:CLOCk] INTernal,<clk_period>
[:FORMat:CLOCk] EXTernal,{LEFifty|GTFifty|GTONe}
```

Example

```
10 DIM Cl$[100]
20 OUTPUT XXX;":FORMAT:CLOCK?"
30 ENTER XXX;Cl$
40 PRINT Cl$
50 END
```

This example queries and prints the current clock settings.

DELay

Command/Query

The DELay command is used to specify the clock out delay. The clock out delay setting allows positioning of the clock with respect to the data. The delay setting that corresponds to zero is uncalibrated and must be measured by the user to determine the basic clock/data timing. Subsequent settings delay the clock approximately 1.3 ns per step.

The query returns the current clock out delay value.

Command syntax

```
:FORMat:DELay<delay_arg>
```

`<delay_arg>` integer from 0 through 9

Query syntax

```
:FORMat:DELay?
```

Returned format

```
[FORMat:DELay]<delay_arg>
```

LABel

Command/Query

The LABel command inserts a new label or modifies the contents of an existing label. If more than 126 labels are specified, and an attempt is made to insert another new label, the last label (bottom label) will be modified.

Only 16 labels may be inserted or modified at a time. If more than 16 labels are specified per command, you will receive an error message.

Pattern generator channels can be assigned to only one label at a time. If duplicate assignments are made, the last channel assignments take precedence.

The second parameter sets the channel polarity. If the polarity is not specified, the last polarity assignment is used. The last parameters assign the active channels for each pod.

Each assignment parameter is a binary encoding of the channel assignments of the pod. The pods are numbered in the same order as they appear in the format menu, with zero representing the left-most pod (pod 5) of the top-most card in the pattern generator module. A "1" in a bit position means that the associated channel in that pod is included in the label. A "0" in a bit position excludes the channel from the label. The minimum value for any pod specification is 0, the maximum value for all pods is 255. A value of 255 includes all channels of a pod assignment. The query must specify a label name and returns the current pod assignments and channel polarity for that label. A maximum of 32 bits can be assigned to a label.

In half channel mode, only the four least significant bits of pod five are used. Pods two and four are not used. Pods one and three use all eight bits. When channel assignments are made in half channel mode, values are required for three pods (1, 3, 5) per card.

Command Syntax

```
:FORMat:LABel <label name>,[<polarity>,]<channel
assignment>, .... <channel assignment>
```

`<label name>` string of up to 6 alphanumeric characters

`<polarity>` polarity of the channel outputs,NEGative or POSitive

`<channel assignment>` a string in one of the following forms:
`'#B01...'` for binary
`'#Q01234567..'` for octal
`'#H0123456789ABCDEF...'` for hexadecimal
`'0123456789...'` for decimal.

Example

Full channel mode, all bits on pods 4 and 5:

```
OUTPUT XXX;":FORMAT:LABEL 'DATA',POS,255,255,0,0,0"
```

Example

Half channel mode, all bits on pods 3 and 5:

```
OUTPUT XXX;":FORMAT:LABEL 'STATUS',NEG,15,255,0"
```

Query Syntax: `:FORMat:LABel? <label name>`

Returned Format: `[:FORMat:LABel] <label name>,<polarity>,<channel assignment>, .... <channel assignment><NL>`

Example

```
10 DIM La$[100]
20 OUTPUT XXX;":FORMAT:LABEL? 'A'"
30 ENTER XXX;La$
40 PRINT La$
50 END
```

This example queries and prints the definition of label 'A'.

MODe

The MODe command is used to specify either FULL or HALF channel output mode. Half channel mode allows a higher output data rate (greater than 100 MHz), but with only 20 channels per card.

Full channel output mode limits the maximum data rate to 100 MHz but allows use of 40 channels per card.

The output mode selection sets the upper limit for the clock rate (see FORMat:CLOCk command).

Command syntax: `:FORMat:MODe{FULL|HALF}`

Query syntax: `:FORMat:MODe?`

Returned format: `[FORMat:MODe]{FULL|HALF}`

Assigning labels in half-channel mode erases previously-assigned labels.

REMove

Command

The REMove is used to delete a single label, or all labels from the format menu. If a label name is specified, it must exactly match a label name currently active in the format menu.

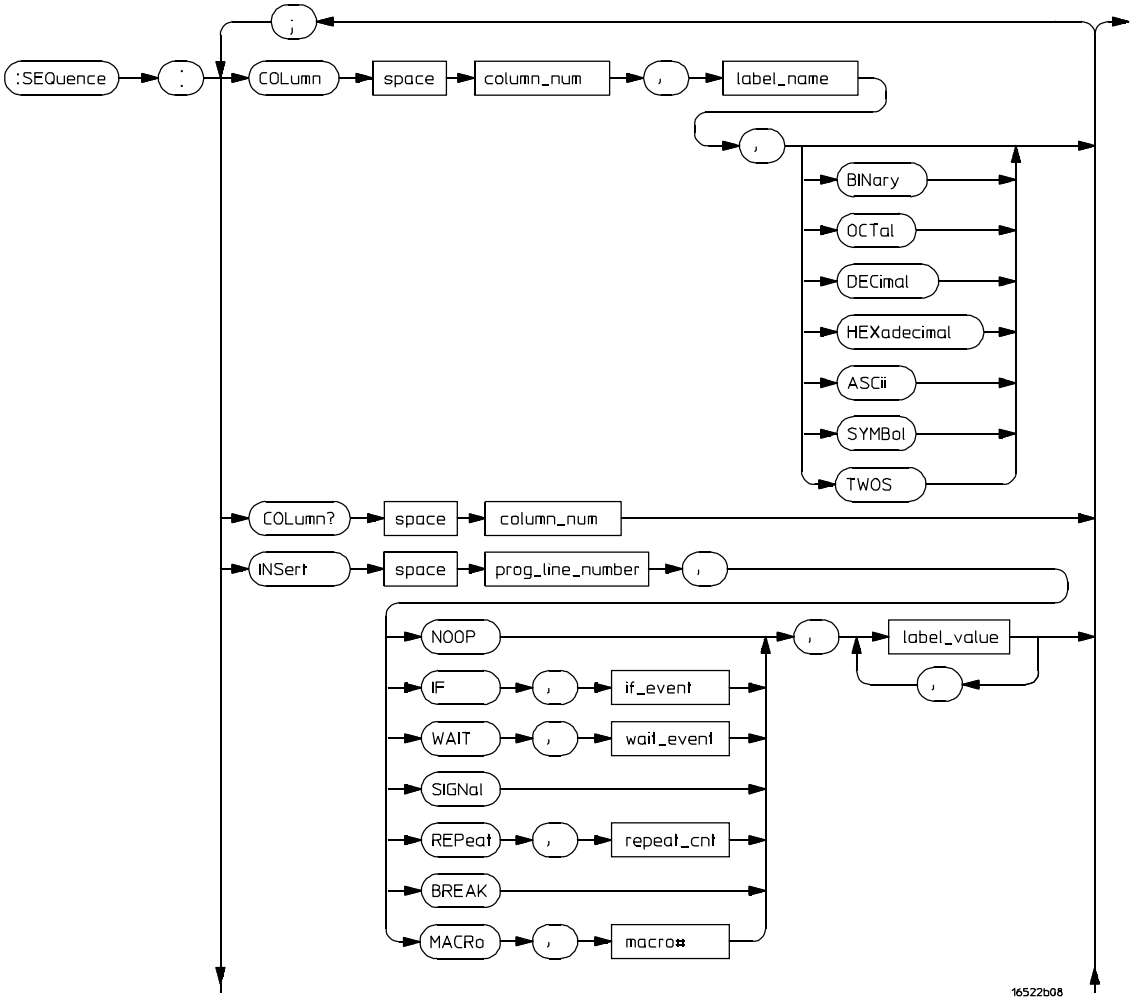
Command Syntax: `:FORMat:REMove {ALL|<label name>}`

`<label name>` a string of up to 6 alphanumeric characters

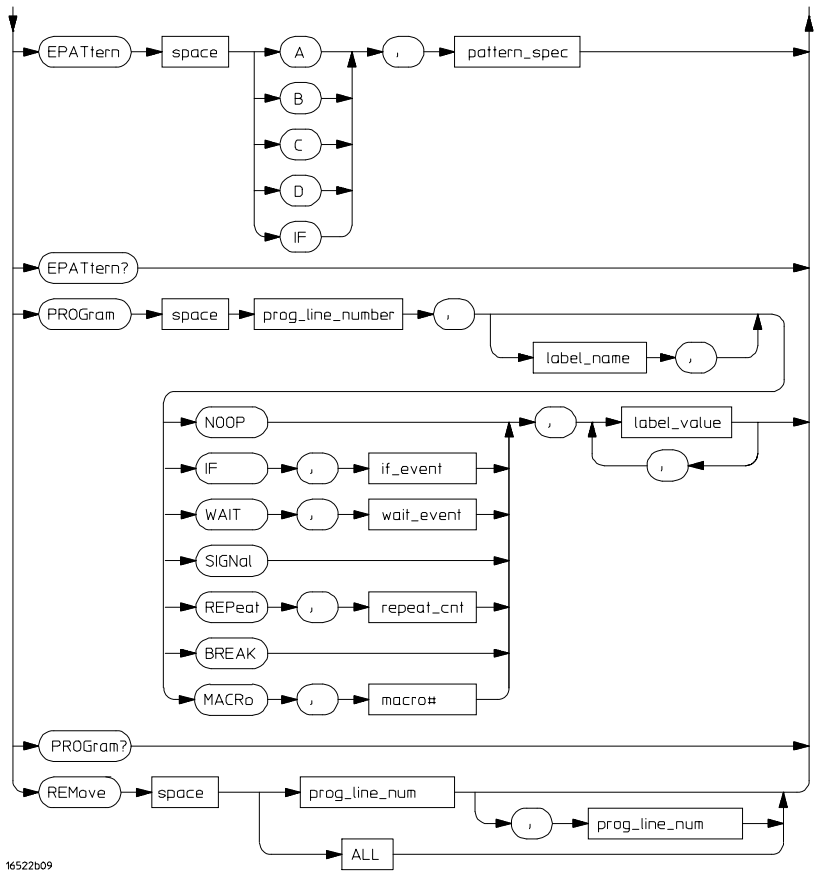
Example

```
OUTPUT XXX;":FORMAT:REMOVE ALL"
```

The commands of the Sequence subsystem allow you to write a pattern generator program using the parameters set in the Format subsystem.



SEQuence Subsystem Syntax Diagram



SEQuence Subsystem Syntax Diagram (cont.)

column_num = *an integer specifying the column that is to receive the* new *label*
label_name = *the label name that is to be removed*
prog_line_num = *an integer specifying the program line number*
label_value = *a string in one of the following forms:*
'#B01...' for binary
'#Q01234567...' for octal
'#H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal
repeat_cnt = *an integer from 1 through 20,000*
macro# = *an integer from 0 to 99*
if_event = *{IF | IMB}*
wait_event = *{A | B | C | D | IMB}*
patter_spec = *an integer from 0 to 255*

COLumn

Command/Query

The COLumn command allows you to reorder the labels in the Sequence and Macro menus and set the numerical base for each label. Label order in the Format menu is not changed when the COLUMN command is used.

The first parameter of the command specifies the column number, followed by a label name and an optional number base. If a number base is not specified, the current number base for the label is used. The instruction field (leftmost column on screen) cannot be moved.

The query must include a column number and returns the label in that column and its base.

Command Syntax:

```
:SEQuence:COLumn <column number>,'<label name>'
[,{BINary|OCTal|DECimal|HEXadecimal|ASCii|SYMBol
|TWOs}]
```

`<column number>` an integer specifying the column that is to receive the new label

`<label name>` a string of up to six alphanumeric characters specifying the label name that is to be moved

Example

```
OUTPUT XXX;":SEQ:COL 1,'A',HEX"
```

Query Syntax:

```
:SEQuence:COLumn? <column number>
```

Returned Format:

```
[SEQuence:COLUMN] <column number>,<label name>,
{BINary|OCTal|DECimal|HEXadecimal|ASCII|
SYMBol|TWOS}
```

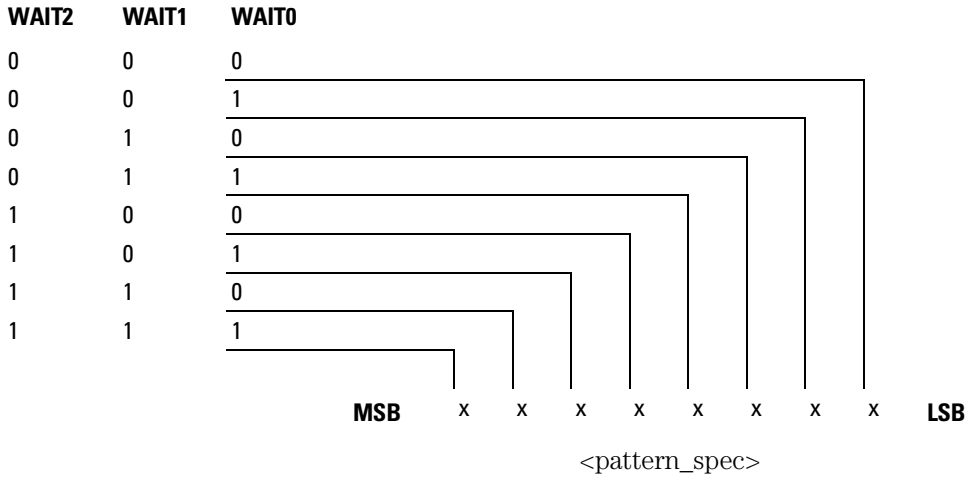
Example

```
10 DIM Co$[100]
20 OUTPUT XXX;":SEQ:COL? 1"
30 ENTER XXX;Co$
40 PRINT Co$
50 END
```

EPATtern

Command/Query

The EPATtern command is used to specify the event patterns used by the WAIT and IF commands. The pattern generator has three external input qualifiers (WAIT2, WAIT1, and WAIT0). There are eight combinations of the three input qualifiers that may be OR'ed together to create an event pattern specification. Mapping of these input qualifier patterns to an event pattern specification is shown below.



<pattern_spec>

The query returns the current pattern specification for the given event.

Command syntax: `:SEQuence:EPATtern { A|B|C|D|IF },<pattern_spec>`

`<pattern_spec>` an integer between 0 and 255 mapping input qualifier combinations as shown above.

Query syntax: `:SEQuence:EPATtern? { A|B|C|D|IF }`

Return format: `[:SEQuence:EPATtern] { A|B|C|D|IF },<pattern_spec>`

See next page for an example.

Example

To specify an event pattern of (0, 1, 0) [Wait2=0, Wait1=1, Wait0=0] use a <pattern_spec> of 4 (0000 0100).

To specify an event pattern of (0, 0, 0) use a <pattern_spec> of 1 (0000 0001).

To specify an event pattern of (0, 1, 1) OR (1, 1, 0) OR (1, 1, 1) use a <pattern_spec> of 200 (1100 1000).

INSert

Command

The INSert command is the basic command used to build a pattern generator sequence. This command is used to insert (or add) a sequence statement after the specified line number.

The first parameter is the line number. The instruction is inserted in the sequence after the specified line number. Sequence lines with instructions other than NOOP cannot be inserted:

- Immediately after the INIT SEQUENCE START line.
- Immediately before or after the start of an IF.
- Immediately before or after the end of an IF.
- Immediately after the MAIN SEQUENCE START line.
- After the MAIN SEQUENCE END line.
- Immediately before the MAIN SEQUENCE END line.

No sequence lines may be inserted between the INIT SEQUENCE END and the MAIN SEQUENCE START lines.

If the line number specified is greater than the MAIN SEQUENCE END line number, the line will be inserted at the last legal location in the main sequence. A legal pattern generator sequence is required to have at least two lines in the main sequence (between MAIN SEQUENCE START and MAIN SEQUENCE END lines).

The second parameter is the instruction for this sequence line. The available instructions are described below

The third parameter is an optional instruction argument. This parameter will only appear when required by a specific instruction.

The last parameter(s) are the data assignments for this line. These assignments are normally made one per label, starting with the left-most column in the display. Note the exception described for the MACRo instruction.

You cannot assign values to more than 16 labels per instruction.

Instructions

NOOP The NOOP instruction means there is no instruction for this line.

BREak The BREak instruction causes the execution of the sequence to stop at this line. Use the RESume command to advance to the next sequence line.

SIGNal The SIGNal instruction is the complement of the WAIT IMB instruction. When the pattern generator encounters a SIGNal instruction, it will output a signal to the internal Intermodule Bus (IMB). This signal is used to trigger the other modules in the Agilent 16500B/C frame that are linked with the IMB.

WAIT The Wait instruction causes the pattern generator to stop and wait for the occurrence of the specified event pattern(s). The event patterns are specified elsewhere (SEQuence: EPATtern command). The event to be waited for by this particular command is specified by the optional instruction argument parameter. Once the specified event occurs, the pattern generator program proceeds to the next state.

Valid wait events are { A | B | C | D | IMB }

IF The IF instruction allows a sequence of program states to occur if a specified condition is true. The IF event pattern can be specified elsewhere (SEQuence:EPATtern command).

The condition to be tested by the IF instruction is specified by the optional instruction argument parameter. If the specified condition is true, the sequence states included in the IF (lines between IF and IF END) are executed. If the condition is not true, the sequence states within the IF are skipped. Valid IF events are {IF | IMB}.

Note that there are clock speed, channel count, and location restrictions on the use of the IF instruction.

REPeat The REPeat instruction allows a group of sequence states to be executed repetitively some number of times. The repeat count is specified in the optional instruction argument parameter.

Inserting a REPeat instruction causes three sequence lines to be generated. The REPeat instruction line, a data line within the body of the repeat, and an END LOOP instruction line.

No data appears in the REPEAT and END LOOP lines. The data specified as part of the remote control command string appears in the body of the repeat loop. Additional data lines can be added to the body of the repeat loop by

inserting lines as needed. The repeat loop is assigned a loop number by the system and is used to connect the limits of the repeat loop.

Note that there are location restrictions on the use of the REPeat instruction.

MACRo# The MACRo# instruction is used to invoke a previously defined user macro. The macro number is part of the instruction string (not the optional instruction argument parameter). If the macro has been defined to use passed-in parameters, those parameter values are passed in via the data value fields. If no parameters are defined, a single dummy parameter must be used ('0'). There is otherwise no data associated with a macro instruction.

Command Syntax

```
:SEQuence:INSert <line_number>,{NOOP|IF,<event>|
WAIT,<event>|SIGNal|REPeat,<count>|BREAK|
MACRo<#>},<data_value>,<data_value>,...
```

`<line_number>` integer where instruction/data will be inserted after

`<event>` { A | B | C | D | IF | IMB }

`<count>` integer repeat count

`<#>` macro number

`<data_value>` a string in one of the following forms:
- `'#B01...'` for binary
- `'#Q01234567...'` for octal
- `'#H0123456789ABCDEF...'` for hexadecimal
- `'0123456789...'` for decimal

Example

```
10 OUTPUT XXX; " :SEQ: INS 248, NOOP, '17', '34', '121'"
20 OUTPUT XXX; " :SEQ: INS 1786, WAIT, A,'17', '34', '121'"
30 OUTPUT XXX; " :SEQ: INS 2652, REPEAT, 26, '17', '34', '121'"
40 OUTPUT XXX; " :SEQ: INS 3166, MACR4, '#HABCD'"
41 !Passes a single parameter to this instance of MACRO #4.
50 OUTPUT XXX; " :SEQ: INS 3186, MACR6, '0'"
51 !Assume no parameter defined for MACRO 6.
```

PROGram

Command/Query

The PROGram command is used to modify an existing pattern generator sequence line.

The first parameter is the line number. The instruction to be modified is at the specified line number. Note that some lines cannot be modified (SEQUENCE START and END) and some instructions can have parameters modified, but the instruction type cannot be changed (REPeat can have the repeat count changed, but it cannot be changed to a NOOP).

The second parameter is an optional label name. The label name allows any data values specified in the command to be assigned starting with the label name rather than defaulting to the first label. This is useful when modifying only a portion of the data for a sequence line.

> You cannot specify more than 16 labels per PROGram command. Use the optional label parameter if the line you want to modify has more than 16 labels.

The third parameter is the instruction. The options for this parameter are described below.

The fourth parameter is an optional instruction argument. This parameter will only appear when required by a specific instruction as described below.

The last parameter(s) are the data assignments for this line. These assignments are normally made one per label, starting with the left-most column in the display.

Note that some instructions cannot be modified. To change the instruction type in these cases, it is necessary to first REMove the line(s) and INSert new lines(s).

The query returns the current contents (instruction and data) for the specified line number.

Instructions

NOOP The NOOP instruction means there is no instruction for this line.

BREak The BREak instruction causes the execution of the sequence to stop at this line. Use the RESume command to advance to the next line sequence.

SIGNal The SIGNal instruction outputs a signal to the internal Intermodule Bus (IMB). This signal is used to trigger the other modules in the Agilent 16500B/C frame that are linked with the IMB.

WAIT The WAIT instruction causes the pattern generator to stop and wait for the occurrence of the specified event pattern(s). The event patterns are set by the SEQuence: EPATtern command. The event to be waited for by this particular command is specified by the optional instruction argument parameter. Once the specified event occurs, the pattern generator program proceeds to the next state.

IF The IF instruction allows a sequence of program states to occur if a specified condition is true. The IF event pattern is specified by the SEQuence:EPATtern command.

The IF and END IF sequence lines cannot be modified other than changing the if condition.

The condition to be tested by the IF instruction is specified by the optional instruction argument parameter. If the specified condition is true, the sequence states include the IF (lines between IF and IF END) are executed. If the condition is not true, the sequence states within the IF are skipped.

Valid IF events are {IF | IMB}.

REPeat The REPeat instruction allows a group of sequence states to be executed repetitively some number of times. The repeat count is specified in the optional instruction argument parameter.

The REPeat and END LOOP sequence lines cannot be modified other than by changing the loop count.

MACRo# The MACRo# instruction invokes a previously defined user macro. It *inserts* the macro, rather than modifying it. The macro number is part of the instruction string (not the optional instruction argument parameter). If the macro has been defined to use passed-in parameters, they are passed in via the data value fields. If there are on parameters associated with the macro, a single dummy parameter must be used ('0'). There is otherwise no data associated with a macro instruction.

Command Syntax

```
:SEQuence:PROGram <line_number>,
[<optional_label>,]{ NOOP | IF,<event> |
WAIT,<event> | SIGNal | REPeat,<count> | BREAK |
MACRo<#> },<data_value>,<data_value>,...
```

`<line_number>` integer where instruction/data will be modified

`<optional_label>` a string of up to 6 alphanumeric characters specifying the label where modification begins.

`<event>` `{A|B|C|D|IF|IMB}`

`<count>` integer repeat count

`<#>` macro number

`<data_value>` a string in one of the following forms:

`'#B01...'` for binary
`'#Q01234567...'` for octal
`'#H0123456789ABCDEF...'` for hexadecimal
`'0123456789...'` for decimal

Query Syntax: `:SEQuence:PROGram? <line_number>`

Note that the returned format of a SEQ:PROG? query is the same instruction text that appears in a front panel listing on the Agilent 16500B/C. This text string cannot be used as part of a command sent to the Agilent 16522A.

Returned Format:

```
{IF (External Pattern = #) | END IF | WAIT
<event> | SIG IMB | START LOOP # REPEAT # TIMES |
END LOOP # | BREAK | MACRO Macro# () | INIT
SEQUENCE START | INIT SEQUENCE END | MAIN
SEQUENCE START | MAIN SEQUENCE END},<data_value>,
<data_value>, ...
```

Example

```
10 OUTPUT XXX; " :SEQ: PROG 248, NOOP, '17', '34', '121'"
20 OUTPUT XXX; " :SEQ: PROG 1786, WAIT, A,'17', '34', '121'"
30 OUTPUT XXX; " :SEQ: PROG 2652, REPEAT, 26, '17', '34', '121'"
40 OUTPUT XXX; " :SEQ: PROG 3166, MACR4, '#HABCD'"
41 ! Passes a single parameter to this instance of MACRO #4.
50 OUTPUT XXX; " :SEQ: PROG 3186, MACR6, '0'"
51 ! Assume no parameter defined for MACRO 6.
```

REMove

Command

The REMove command allows you to remove one or several lines from the pattern generator program. If only one parameter number is given, that line number is deleted. If two numbers are given, the range of lines between those two values inclusive is deleted. The command REMove ALL deletes the entire program.

Command Syntax:

```
SEQuence:REMove{ <program line number[,<program line range>]|ALL>}
```

`<program line number>` an integer specifying the program line to be removed

`<program line range>` an integer specifying the last line number in a range of lines to remove.

Example

```
OUTPUT XXX;":SEQ:REM 1,4"
```

The commands of the MACRo subsystem allow you to write and edit macros for use in the pattern generator program. Up to 100 macros may be called into the main listing program. The macros are labeled Macro0 through Macro99.

Macro0 is always available (initial contents are START/END lines only). All other macros are created whenever a MACRo<#> subheader that is not yet defined is used. The new macro will then appear on all macro lists until a MACRo<#>:REMove command is issued.

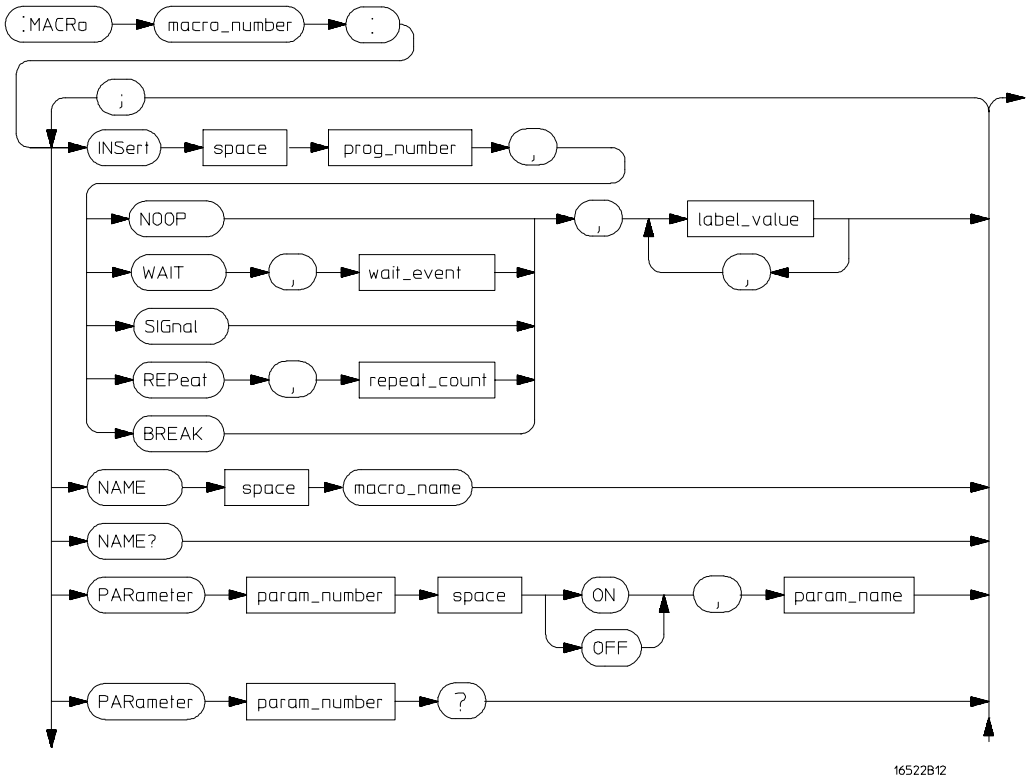
A macro can be named (MACRo<#>:NAME command) but cannot be referenced by remote control commands using that name.

The SEQuence:COLumn command is used to define the ordering of the sequence display listing. Macro display listings will appear in the same order as the main sequence. Changing the display while on a macro listing will also affect the main sequence when you return to that display listing.

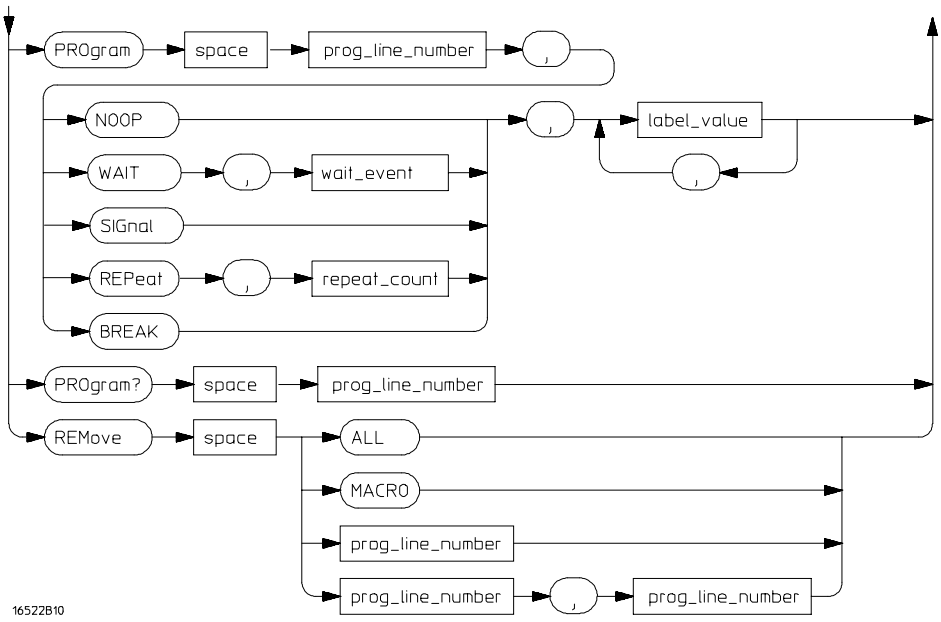
The SEQuence:EPATtern command is used to define event patterns that are shared by both the main sequence and all macros. Changing an event pattern definition for use by a single macro will change its definition for all other macros and the main sequence.

The command REMove ALL can be used to totally clear the contents of a macro, but it does not remove the macro from the macro list. The macro is still accessible from the sequence, but the macro consist of only two lines.

The command REMove MACRo can be used to totally remove all contents of a macro as well as any external reference to that macro. Note that while Macro0 can be totally cleared, it cannot be removed from the macro list.



MACRo Subsystem Syntax Diagram



MACRo Subsystem Syntax Diagram (cont.)

prog_line_num = *an integer specifying the program line number*
macro_name = *character string up to 6 characters in length*
macro_number = *an integer 0 through 99 specifying macro to act on*
param_name = *character string up to 6 characters in length*
param_number = *an integer 0 through 9*
repeat_count = *an integer from 1 through 20000*
wait_event = *{ A | B | C | D | IMB }*
label_name = *character string up to 6 characters in length*
label_value = *data entry in one of the following forms:*
'#B01...' for binary
'#Q01234567...' for octal
'#H012345679ABCDEF...' for hexadecimal
'0123456789...' for decimal
PARameter<#> for passed in macro parameter (# = 0 through 9)

INSert

Command

The INSert command is the basic command used to build a pattern generator macro. This command is used to insert (or add) a macro statement after the specified line number.

The first parameter is the line number. The instruction and/or data will be inserted in the macro after the specified line number. You cannot insert a line just before the last data row. Macro lines cannot be inserted after the MACRO END line.

If the line number specified is greater than the MACRO END line number, the line will be inserted at the last legal location in the macro.

The second parameter is the instruction for this macro line. The available instructions are described below

The third parameter is an optional instruction argument. This parameter will only appear when required by a specific instruction.

The last parameter(s) are the data assignments for this line. These assignments are normally made one per label, starting with the left-most column in the display. In addition to the normal data values, parameters passed in with a macro call can be inserted within the body of the macro.

Instructions

NOOP The NOOP instruction means there is no operation for this line.

BREak The BREak instruction causes the execution of the sequence to stop at this line. Use the RESume command to advance to the next macro line.

SIGNal The SIGNal instruction outputs a signal to the internal Intermodule Bus (IMB). This signal is used to trigger the other modules in the Agilent 16500B/C frame that are linked with the IMB.

WAIT The WAIT instruction causes the pattern generator to stop and wait for the occurrence of the specified event pattern(s). The event to be waited for by this particular command is specified by the optional instruction argument parameter. Once the specified event occurs, the pattern generator program proceeds to the next state.

Valid wait events are { A | B | C | D | IMB }. Their patterns are set using the SEQuence: EPATtern command.

REPeat The REPeat instruction allows a group of states to be executed repetitively some number of times. The repeat count is specified in the optional instruction argument parameter.

Inserting a REPeat instruction causes three lines to be generated: the REPeat instruction line, a data line within the body of the repeat, and an END LOOP instruction line. No data appears in the REPEAT and END LOOP lines. The data specified as part of the remote control command string appears in the body of the repeat loop. Additional data lines can be added to the body of the repeat loop by inserting lines as needed. The repeat loop is assigned a loop number by the system and is used to connect the limits of the repeat loop.

Command Syntax

```
:MACRo<m#>:INSert <line_number>, { NOOP |
WAIT,<event> | SIGNal | REPeat,<count> | BREAK }
,<data_value>,<data_value>,...
```

`<line_number>` integer which line instruction/data will be inserted after

`<event>` `{ A | B | C | D | IMB }`

`<count>` integer repeat count

`<m#>` macro number (integer 0 through 99)

`<p#>` parameter number (integer 0 through 9)

`<data_value>` a string in one of the following forms:

`'#B01...'` for binary
`'#Q01234567...'` for octal
`'#H0123456789ABCDEF...'` for hexadecimal
`'0123456789...'` for decimal
`PARameter<p#>`

Example

```
OUTPUT XXX;":MACRO4:INSERT 3, BREAK, PAR1, '13'"
```

NAME

Command/Query

The NAME command is used to specify a name for a macro. This name will then appear in the front panel lists and displays in place of the more generic "Macro #" string.

The name cannot be used to reference the macro in programs. It is intended for use as a means to clarify or document sequence listings and displays.

The query returns the user-defined macro name.

Command syntax: `:MACRo<#>:NAME <macro_name>`

`<macro_name>` a string up to six alphanumeric characters in length

`<#>` macro number (integer 0 through 99).

Query syntax: `:MACRo<#>:NAME?`

Return format: `[:MACRo<#>:NAME] <macro_name>`

PARameter

Command/Query

The PARameter command is used to enable and name parameters for a macro. The parameter name is optional, and if used, is for use on displays and listings only. When a parameter is enabled, macro calls from the sequence can pass values to the macro. These values can then be used as data values in the body of the macro.

The query returns the current status of a parameter and its name.

Command syntax: `:MACRo<m#>:PARameter<p#> { ON | OFF }[,<name>]`

`<m#>` macro number (integer 0 through 99)

`<p#>` parameter number (integer 0 through 9)

`<name>` string up to six alphanumeric characters in length

Query syntax: `:MACRo<m#>:PARameter<p#>?`

Returned format: `[:MACRo<m#>:PARameter<P#>] { ON | OFF },<name>`

PROGram

Command/Query

The PROGram command is used to modify an existing pattern generator macro line.

The first parameter is the line number of the instruction to be modified. Note that some lines cannot be modified (MACRO and MACRO END) and some instructions can have parameters modified. The instruction type cannot be changed (REPeat can have the repeat count changed, but it cannot be changed to a NOOP).

The second parameter is an optional label name. The label name allows any data values specified in the command to be assigned starting with the label name rather than defaulting to the first label. This is useful when modifying only a portion of the data for a macro line.

> You can only modify 16 labels per PROGram command. To modify more than 16 labels, use the optional label name parameter.

The third parameter is the instruction. The options for this parameter are described below.

The fourth parameter is an optional instruction argument. This parameter will only appear when required by a specific instruction as described below.

The last parameter(s) are the data assignments for this line. These assignments are normally made one per label, starting with the left-most column in the display. In addition to the normal data values, parameters passed in with a macro call can be inserted within the body of the macro. Specifying more than 16 data assignments will cause a command error.

Note that some instructions cannot be modified. To change the instruction type in these cases, it is necessary to first REMove the line(s) and INSert new lines(s).

The query returns the current contents (instruction and data) for the specified line number.

Instructions

NOOP The NOOP instruction means there is no operation for this line.

BREak The BREak instruction causes the execution of the macro to stop at this line. Use the RESume command to advance to the next line macro.

SIGNal The SIGNal instruction outputs a signal to the internal Intermodule Bus (IMB). This signal is used to trigger the other modules in the Agilent 16500B/C frame that are linked with the IMB.

WAIT The WAIT instruction causes the pattern generator to stop and wait for the occurrence of the specified event pattern(s). The event to be waited for by this particular command is specified by the optional instruction argument parameter. Once the specified event occurs, the pattern generator program proceeds to the next state.

Valid WAIT events are { A | B | C | D | IMB }. Their patterns are set using the SEQuence: EPATtern command.

REPeat The REPeat instruction allows a group of macro states to be executed repetitively some number of times. The repeat count is specified in the optional instruction argument parameter.

The REPeat and END LOOP sequence lines cannot be modified other than to change the loop count.

Command Syntax

```
:MACRo<m#>:PROGram <line_number>,
[<optional_label>,]{ NOOP | WAIT,<event> | SIGNal
| REPeat,<count> | BREAK }
,<data_value>,<data_value>,...
```

`<line_number>` integer specifying the line of instruction/data to be modified

`<optional_label>` a string of up to six characters specifying a label

`<event>` `{ A | B | C | D | IMB}`

`<count>` integer repeat count

`<m#>` macro number (integer 0 through 99)

`<p#>` parameter number (integer 0 through 9)

`<data_value>` a string in one of the following forms:

`'#B01...'` for binary
`'#Q01234567...'` for octal
`'#H0123456789ABCDEF...'` for hexadecimal
`'0123456789...'` for decimal
`PARameter<p#>`

Query Syntax:

```
:MACRo<#>:PROGram? <line_number>
```

Returned Format:

```
[:MACRo<#>:PROGram]  <line_number>, { NOOP | WAIT
<event> | SIG IMB | BREAK  | MACRO END | START
LOOP # REPEAT # TIMES | END LOOP #  | MACRO
Macro# ()  },<data_value>, <data_value>, ...
```

REMove

Command

The REMove allows you to remove one or several lines from the macro. If only one parameter is given, only that line is deleted. If two numbers are specified, the range of lines between those values, inclusive, is deleted.

The command REMove ALL can be used to totally clear the contents of a macro, but it does not remove the macro from the macro list. This means the macro is still accessible from the sequence, but the macro consists of only two lines.

The command REMove MACRo can be used to totally remove all contents of a macro as well as any external reference to the macro. Note that while Macro0 can be totally cleared, it cannot be removed from the macro list.

Command Syntax:

```
:MACRo<macro number>:REMove {<program line
number>[,<program line number>]|ALL|MACRo}
```

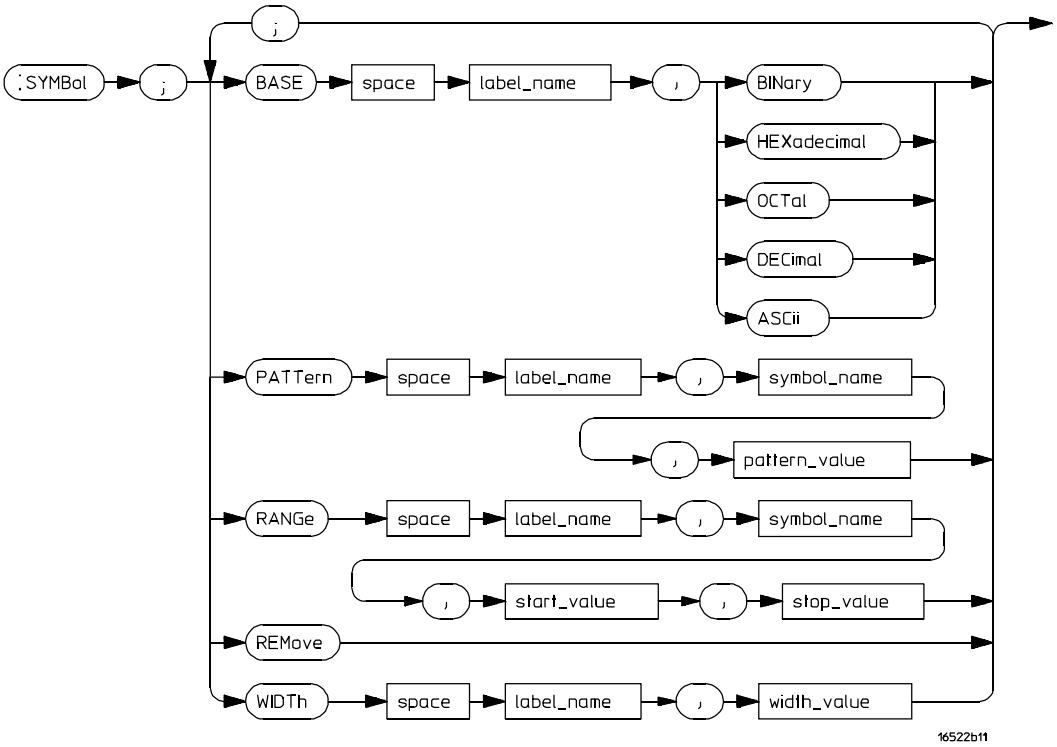
`<macro number>` an integer, 0 through 99

`<program line>` an integer specifying the program line to be removed

Example

```
OUTPUT XXX;":MACRO1:REM 1,3"
```

The SYMBol subsystem contains the commands that allow you to define symbols on the controller and download them to the Pattern Generator module.



SYMBol Subsystem Syntax Diagram

<label_name> = *string of up to 6 alphanumeric characters*
<symbol_name> = *string of up to 16 alphanumeric characters*
<pattern_value> = *string of one of the following forms:*
'#B01X...' for binary
'#Q01234567X..' for octal
'#H0123456789ABCDEFX...' for hexadecimal
'0123456789...' for decimal
<start_value> = *string of one of the following forms:*
'#B01...' for binary
'#Q01234567..' for octal
'#H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal
<stop_value> = *string of one of the following forms:*
'#B01... for binary
'#Q01234567..' for octal
"#H0123456789ABCDEF..." for hexadecimal
'0123456789...' for decimal
<width_value> = *integer from 1 to 16*

BASE

Command

The BASE command sets the base in which symbols for the specified label will be displayed in the symbol menu. It also specifies the base in which the symbol offsets are displayed when symbols are used.

Note that BINary is not available for labels with more than 20 bits assigned. In this case the base will default to HEXadecimal.

Command Syntax: `:SYMBol:BASE <label_name>,<base_value>`

`<label_name>` string of up to 6 alphanumeric characters

`<base_value>` `{BINary | HEXadecimal | OCTal | DECimal | ASCii }`

Example

```
OUTPUT XXX;":SYMBol:BASE 'DATA',HEXadecimal"
```

PATTern

Command

The PATTern command allows you to specify a symbol for a pattern on the specified label. The pattern may contain "don't cares" in the form of XX...X's.

Command Syntax: `:SYMBol:PATTern<label_name>,<symbol_name>,<pattern_value>`

`<label_name>` string of up to 6 alphanumeric characters

`<symbol_name>` string of up to 16 alphanumeric characters

`<pattern_value>` string of one of the following forms:

`'#B01X...'` for binary
`'#Q01234567X..'` for octal
`'#H0123456789ABCDEFX...'` for hexadecimal
`'0123456789...'` for decimal

Example

```
OUTPUT XXX;":SYMBol:PATTern 'STAT', 'MEM_RD','#H01XX'"
```

RANGe

Command

The RANGe command allows you to create a symbol for a range of values on a label. Note that Don't Cares are not allowed in range symbols.

Command Syntax:

```
:SYMBol:RANGe<label_name>,<symbol_name>,
<start_value>,<stop_value>
```

`<label_name>` string of up to 6 alphanumeric characters

`<symbol_name>` string of up to 16 alphanumeric characters

`<start_value>`
`<stop_value>` string in one of the following forms:
`'#B01...'` for binary
`'#Q01234567..'` for octal
`'#H0123456789ABCDEF...'` for hexadecimal
`'0123456789...'` for decimal

Example

```
OUTPUT XXX;":SYMBol:RANGe 'STAT',
'IO_ACCESS','#H0000','#H000F'"
```

REMove

Command

The REMove command deletes all symbols from the symbol menu.

Command Syntax: `:SYMBol:REMove`

Example

```
OUTPUT XXX;":SYMBol:REMove"
```

WIDTh

Command

The WIDTh command specifies the number of characters displayed when symbols are used.

Note that the WIDTh command does not affect the displayed length of the symbol value.

Command Syntax: `:SYMBol:WIDTh <label_name>,<width_value>`

`<label_name>` string of up to 6 alphanumeric characters

`<width_value>` integer from 1 to 16

Example

```
OUTPUT XXX;":SYMBol:WIDTh 'DATA',9 "
```

The DATA and SETup commands are system commands that allow you to send and receive instrument configuration, setup and program data to and from a controller in block form. This is useful for saving block data for re-loading the pattern generator. This chapter explains how to use these commands.

The block data for the DATA command is broken into byte positions and descriptions. The SETup command block data is not described in detail. No changes should be made to the "config" section of the block data.

Definition of Block Data

Block data is made up of a block length specifier and a variable number of sections.

```
<block length specifier><section 1>...<section N>
```

`<block length specifier>` `#8<length>`

`<length>` the total length of all sections in byte format (must be represented with 8 digits)

Example

If the total length of the block (all sections) is 14506 bytes, the block length specifier would be "#800014506" since the length must be represented with 8 digits.

Sections consist of a section header followed by the section data as follows:

`<section>` `<section header><section data>`

`<section header>` 16 bytes total: 10 bytes for the section name, 1 byte reserved (always 0),
1 byte for the module ID code (25 for pattern generator),
4 bytes for the length of the data in bytes

`<section data>` The section data format varies for each section and may be any length.

Note that the total length of a section is 16 (for the section header) plus the length of the section data. Thus, when calculating the length of a block of configuration data, don't forget to add the length of the headers.

Example

```
10 DIM Block$[32000]   !allocate enough memory for block data
20 DIM Specifier$[2]
30 OUTPUT XXX;"EOI ON"
40 OUTPUT XXX;"SYSTEM:HEAD OFF"
50 OUTPUT XXX;"SELECT 1"   !select module
60 OUTPUT XXX;"SYSTEM:DATA?   !send the data query
70 ENTER XXX USING"#,2A";Specifier$   !read in #8
80 ENTER XXX USING"#,8D",Blocklength   !read in block length
90 ENTER XXX USING"-K",Block$   !read in data
```

SYSTem:DATA

The DATA command is used to send and receive the pattern generator main program listings and the macro listings. The complete pattern generator data block consists of two sections not counting the SYMBOL section. The sections are:

Section 1 "DATA"
Section 2 "MACROS"

Command Syntax: `:SYSTem:DATA <block data>`

Query Syntax: `:SYSTem:DATA?`

Returned Format: `[:SYSTem:DATA] <block data><NL>`

Section 1 "DATA"

The Main Program section contains the program listing data in binary form. The length of this section depends on the length of the program listing and the number of expansion cards connected to the master card.

Section 2 "MACROS"

The "MACROS" section contains all the program listing for all the macros. The length of this section varies depending on the length of the macro listings and the number of expansion cards connected to the master card.

To load data into the pattern generator in ASCII form, use the mainframe command :MMEM:LOAD.
See Chapter 6 for information on creating ASCII files for the pattern generator.

SYSTem:SETup

The SETup command for the pattern generator module is used to configure system parameters, such as the pod and bit assignment, clock rates, and output mode by loading saved configurations.

The "CONFIG" section consists of 4082 bytes of information which fully describe the main parameters for the pattern generator. The total length of the section is 4082 bytes (recall that the section header is 16 bytes).

The data in this section of the block should not be changed to ensure proper pattern generator operation.

Command Syntax: `:SYSTem:SETup <block data>`

Query Syntax: `:SYSTem:SETup?`

Returned Format: `[:SYSTem:SETup] <block data><NL>`

4

Servicing the Agilent 16522A

Testing Interval

Periodic performance verification (full calibration) is not required for the Agilent 16522A module. Because there are no specifications for the Agilent 16522A, testing against specifications does not apply.

To Use the Flowcharts

Flowcharts are the primary tool used to isolate defective assemblies. The flowcharts refer to other tests to help isolate the trouble. The circled letters on the charts indicate connections with the other flowcharts. Start your troubleshooting at the top of the first flowchart.

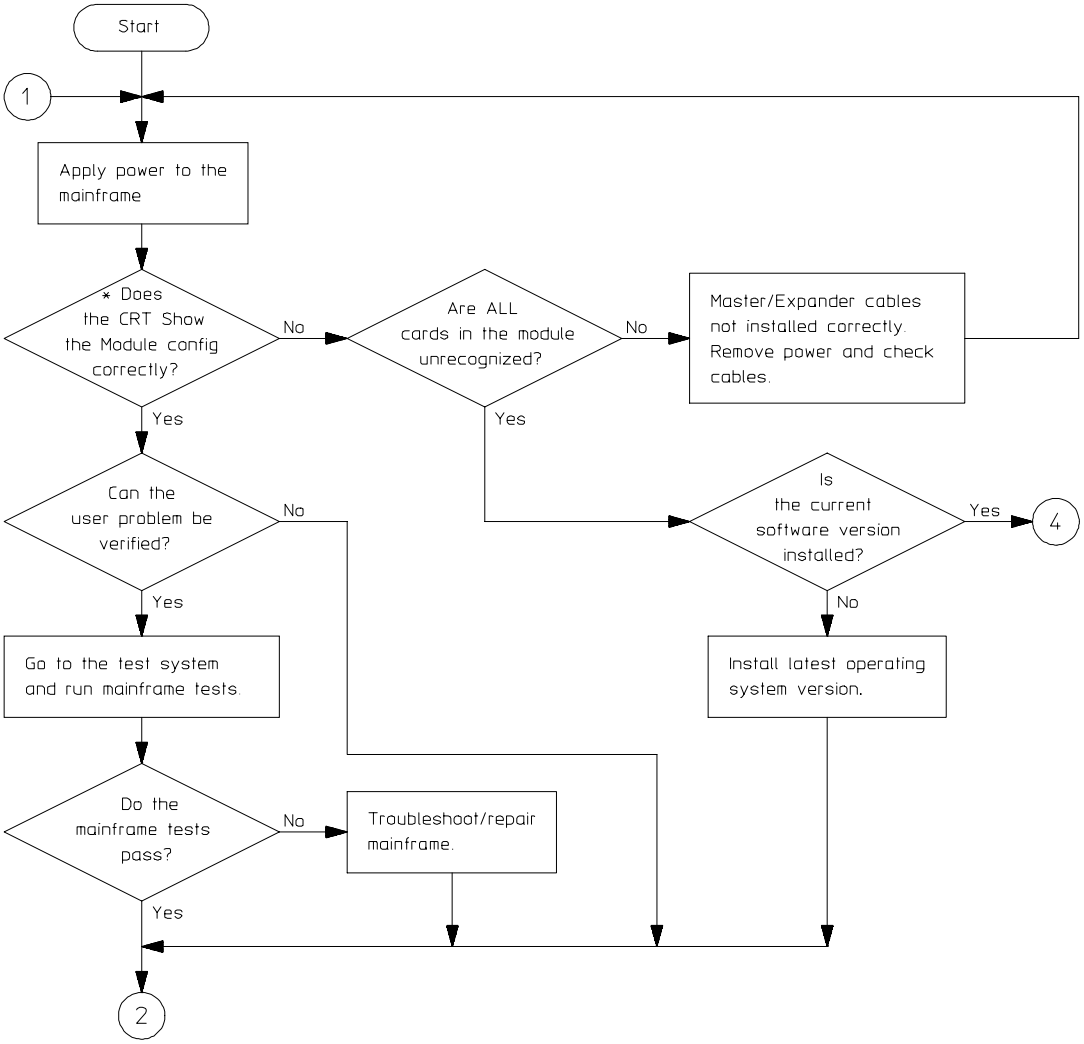
With Agilent 16500B Mainframe

Before starting the troubleshooting, ensure that operating system v3.05 or later is installed on the mainframe. To check the operating system version number, enter the System Test menu.

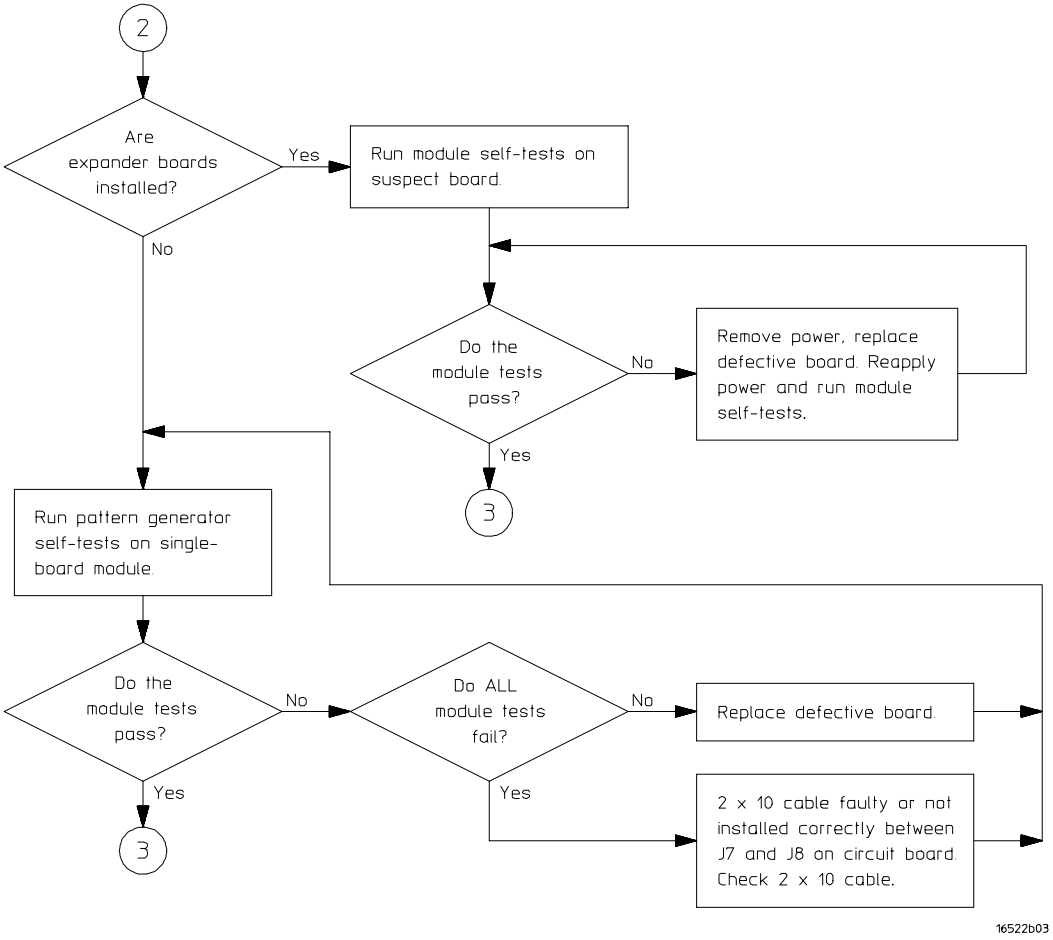
If v3.05 or later is not loaded, obtain a copy of the latest operating system software and install it on the logic analyzer hard drive. After installation, recycle power on the mainframe.

With Agilent 16500C Mainframe

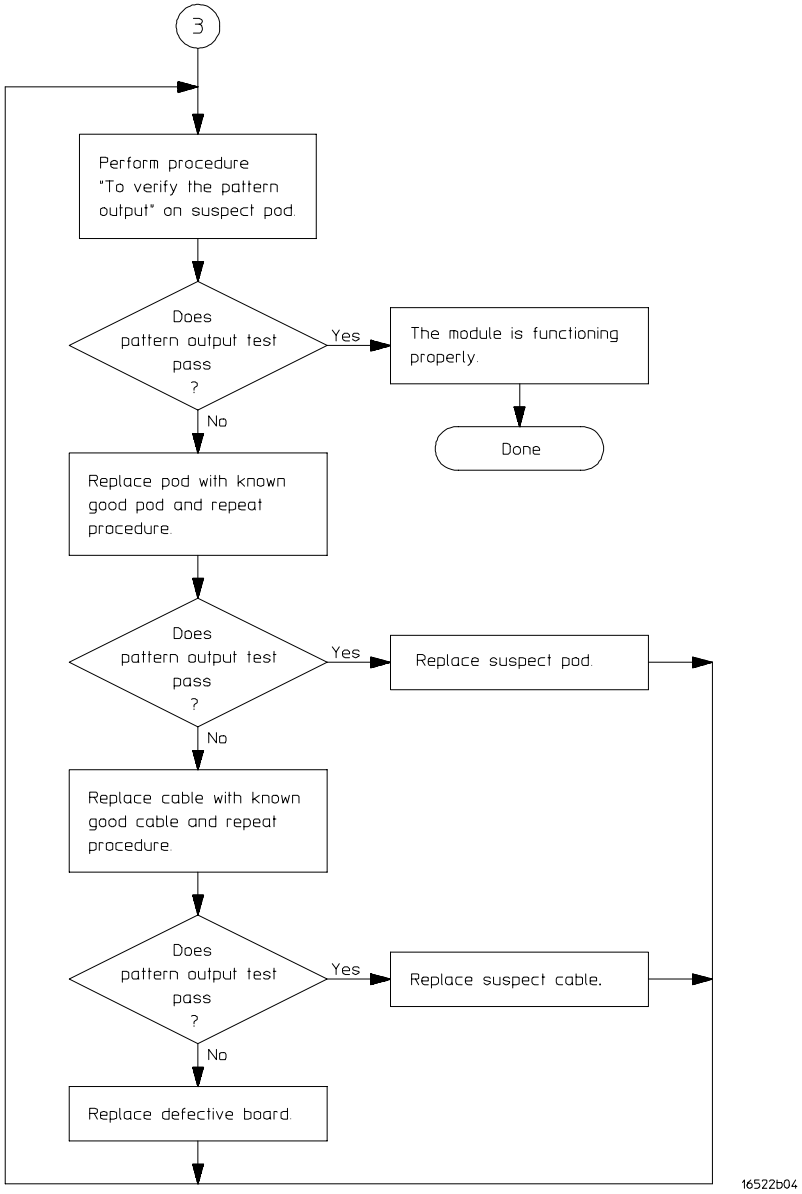
The Agilent 16522A works with all operating system versions on the Agilent 16500C.



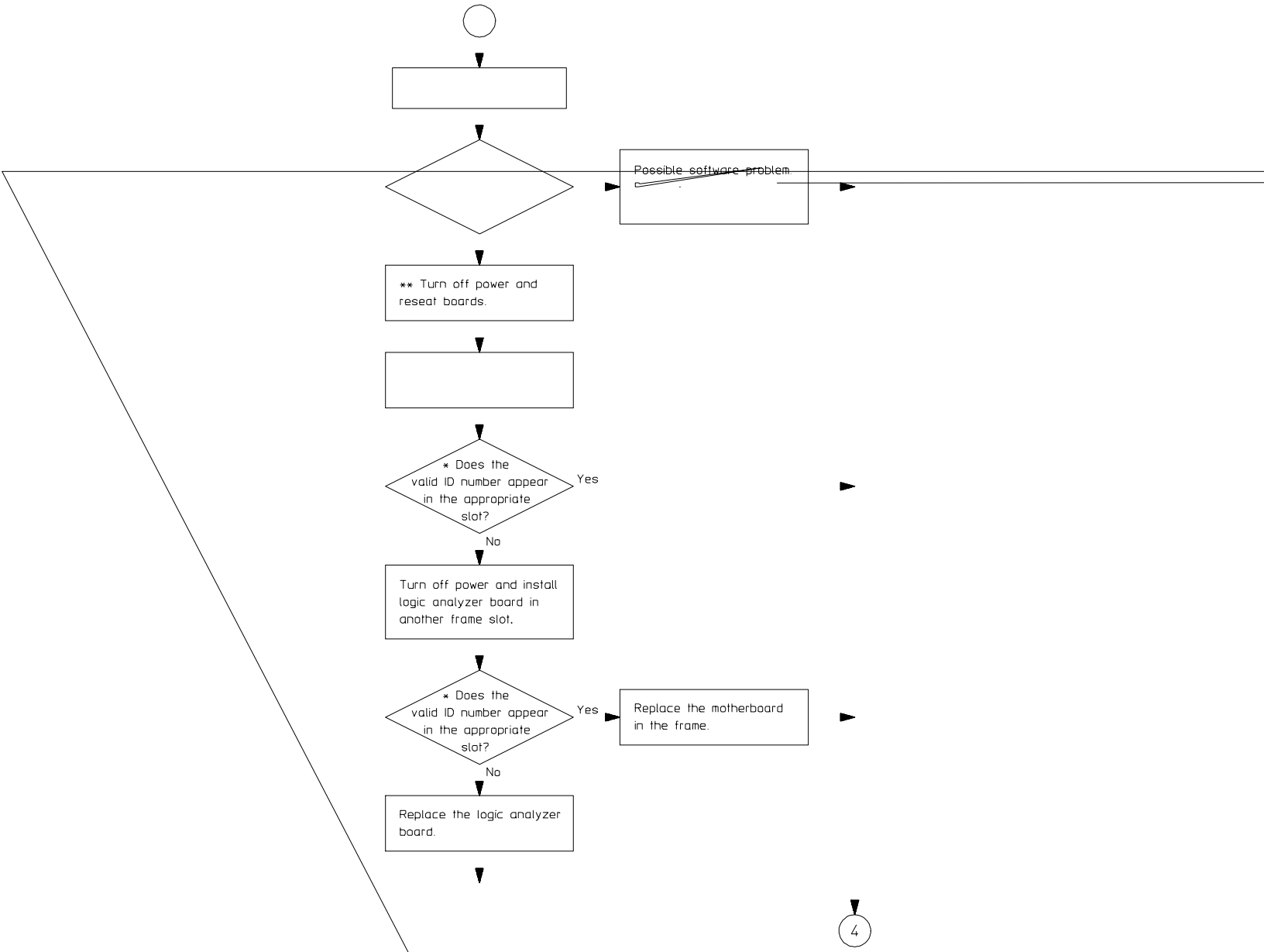
Troubleshooting Flowchart 1



Troubleshooting Flowchart 2



Troubleshooting Flowchart 3



Troubleshooting Flowchart 4Servicing the Agilent 16622A4

The following procedures test the performance of the 16522A pattern generator. The PASS or FAIL status of a test is reported in the individual test items on the display of the 16500B/C mainframe.

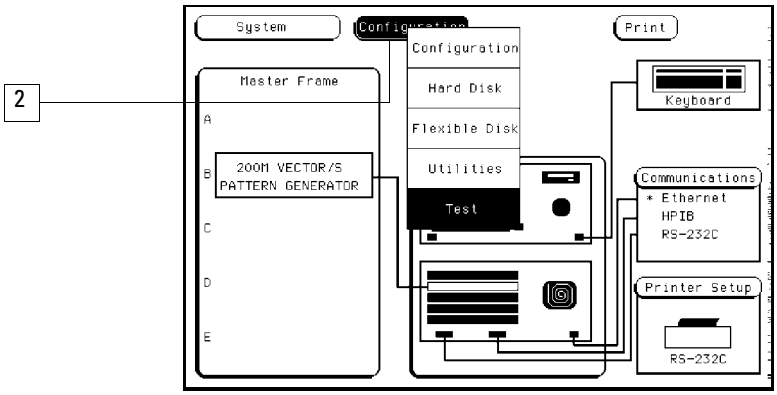
When running in the 'All Tests' mode the status is updated after all tests are complete. Tests may be run individually from the front panel by selecting the test item. When a test is run individually from the front panel the number of runs and failures of a test are displayed for the user. In the individual test mode, when a test fails an error message is displayed which contains an integer value in hexadecimal format for each card in the module. The integer value is used as a diagnostic tool for isolating problems on the board. The actual definition of the bit locations in the integer value are explained with each test.

Fixed output patterns are also available to the user from the front panel. When the user selects the 'Output Pattern' item on the display, a menu appears that allows the user to select the frequency mode of the output pattern (50MHz, 100MHz, 200MHz) and the type of pattern to output. Selecting a pattern item in this menu will load and run the desired pattern. Patterns are run at the maximum clock rate for the current frequency mode. Changing the frequency mode, pressing the 'Stop' item, or closing the menu will stop the output pattern.

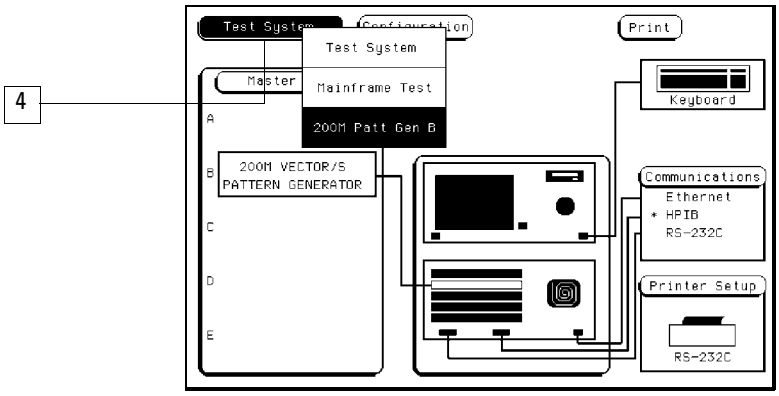
To run self-tests

Self-tests identify correct operation of major functional areas. The self-tests consist of four functional tests and a pattern output routine. You run all self-tests without accessing the instrument's interior. If a self-test fails, the troubleshooting flowcharts instruct you to change a card or a cable.

1. Disconnect all inputs, then turn on the power switch.
2. In the System Configuration menu, touch Configuration, then touch Test.



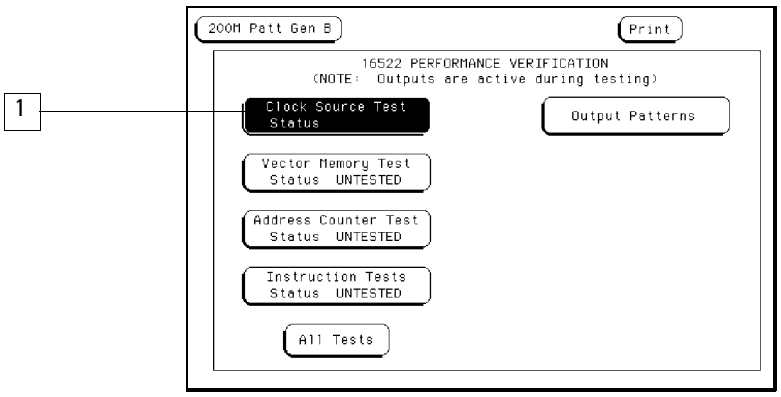
3. In the Test menu, touch the box labeled Touch Box to Load Test System.
4. In the Test System menu, touch Test System. Select 200M Patt Gen module to be tested.



To run performance verification tests

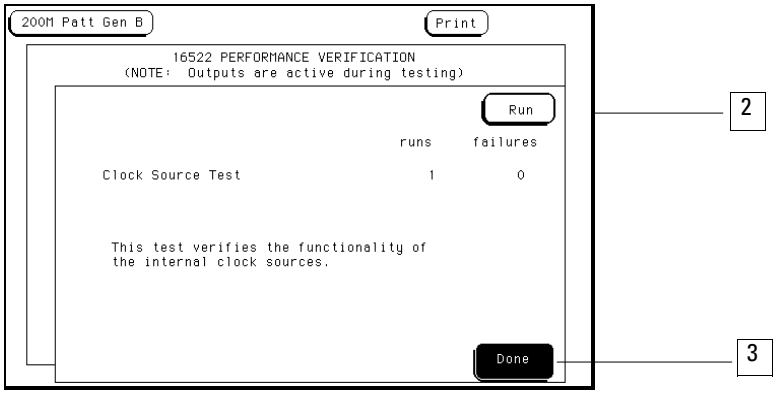
1 In the Performance Verification menu, touch Clock Source Test.

You can run all tests at one time (except the Output Patterns routine) by touching All Tests. To see more details about each test, you can run each test individually. This example shows how to run the Clock Source Test. The Vector Memory Test, Address Counter Test, and Instruction Tests are run in a similar manner.



2 In the Clock Source Test menu, touch Run. The test runs one time, and the screen shows the result.

To run a test continuously, touch and hold your finger on Run. Drag your finger to Repetitive, then lift your finger. Touch Stop to stop Run Repetitive.



3 When the test is finished, touch Done. Then, perform the Vector Memory Test, Address Counter Test, and Instruction Tests.

4 Continue to the next procedure to verify the pattern output to ensure the output driver circuitry is functioning.

5 If any of the tests do not pass, contact an Agilent Technologies Sales Office for repair.

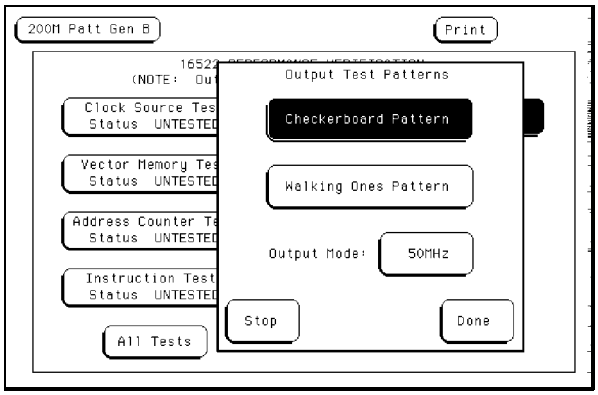
To verify pattern output

Table 4-1 **Equipment Required**

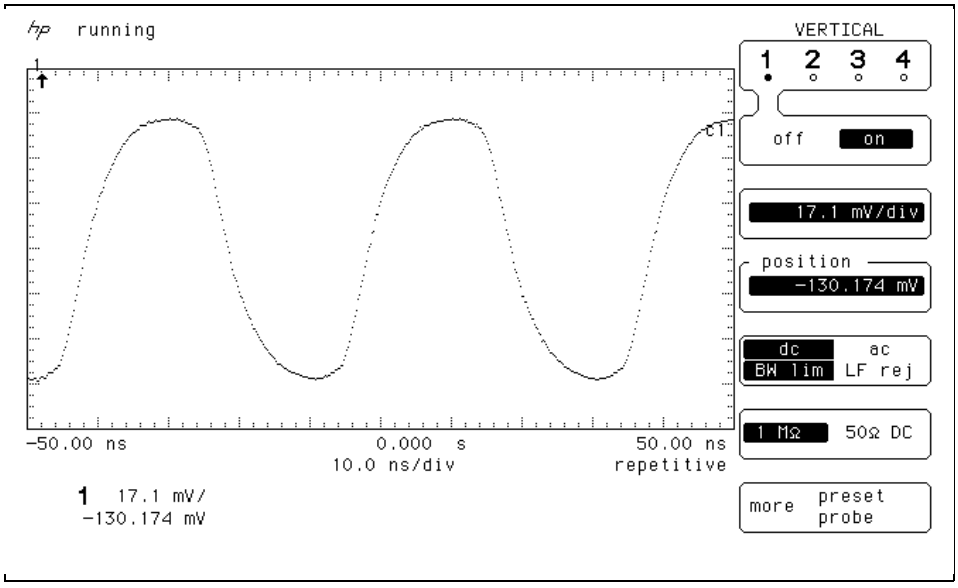
Equipment	Critical Specification	Recommended Model/Part
Oscilloscope	≥ 500 MHz Bandwidth	Agilent 54522A
Probe	500 MHz Bandwidth	Agilent 10441A
Output Data Pod	no substitute	10460A - series

1 Connect one of the 10460-series data pods to the end of the pattern generator Pod 1 cable.

2 Touch Output Patterns. In the pop-up menu, touch Checkerboard Pattern.



3 Using an oscilloscope, verify the existence of logic-level transitions by touching the oscilloscope probe to each channel of Data Pod 1 and doing an Autoscale. The signal levels that appear on the oscilloscope display should correspond with the logic levels represented by the 10460-series pod being used.



4 Repeat step 3 for each of the remaining four data pods.

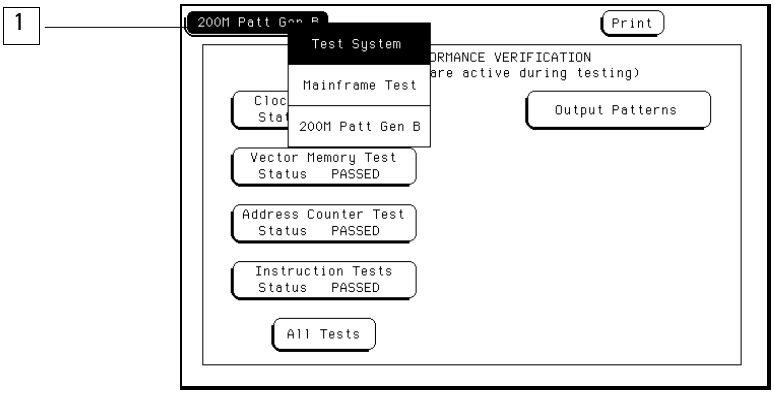
5 Connect one of the 10460-series clock pods to the end of the pattern generator clock cable.

6 Using the oscilloscope as in step 3, verify the existence of logic-level transitions by touching the oscilloscope probe to each clock output of the clock pod.

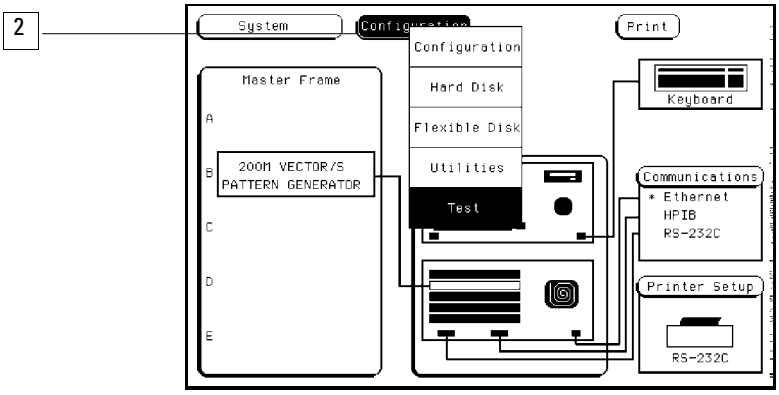
7 In the pattern generator Output Patterns menu, touch Stop, then touch Done to exit the menu.

To exit the test system

1 To exit the Performance Verification menu, touch 200M Patt Gen, then select Test System.



2 To exit the test system, touch Configuration, then select Test. Touch the box labeled Touch Box to Exit Test System.



3 If you are performing the self-tests as part of the troubleshooting flowchart, return to the troubleshooting flowcharts.

The following section contains a description of each of the the Agilent 16522A self tests. In a multi-card master/expander module, the tests are performed on all cards unless otherwise indicated.

Clock Source Test

The Clock Source Test checks that the internal clock sources are functioning by verifying the presence of a given clock source. The test toggles each clock source in the following fashion. First the board is stopped and outputs are disabled. Module RAM is loaded with zeros, then the module is placed in the respective mode for the given clock and the clock source is selected. The module is then started and the main status checked to see that the pipeline is running. The board will then be stopped and the status checked to see that the pipeline did stop.

Passing the Clock Source Test implies that the module internal clock sources are functioning properly, and that the other dependent subcircuits of the module respond to the clock signal.

DIAGNOSTIC INTEGER VALUE:

This test is only valid for signals on the master board of the configuration. The values returned from the expanders will be zero. The integer returned will have the following bit format:

BIT #:	15, 14,13, 12	11, 10, 9, 8	7, 6, 5, 4	3, 2, 1, 0
	200M clk	100M clk	50M clk	PLD clk

Each nibble of the output corresponds to one of the clock sources. The bit pattern of each nibble has the following definition:

0 — passed
1 — failed to run
2 — failed to stop
3 — failed to both run and stop

Vector Memory Test

The Vector Memory Test does a first order check of the functionality of module RAM. The first pass of the test will load the entire RAM with 0x0000. The software will step the clock enough times to output one page worth of data. At each clock a test read port for each RAM IC in the module will be checked and verified for all 0s.

The second pass of the test will load all the RAMS with 0xFFFF and then check using the same technique as in the first pass, verifying for all Fs.

The third pass loads memory with an alternating 0x5555 and 0xAAAA checkerboard pattern. Again the test checks the data in the same fashion as in the first pass.

Passing the Vector Memory Test implies that each memory location in module RAM can store a logic 1 or 0. Passing the test also implies that the CPU interface is functioning and can properly affect control over the memory and memory addressing.

DIAGNOSTIC INTEGER VALUE:

This test checks the RAM of the entire module so the diagnostic integer will be valid for the master card and all expander-configured cards. The returned integer for a particular card has the following format:

BIT #:	15, 14	13, 12, 11, 10, 9, 8, 7, 6	5, 4, 3, 2, 1, 0
	Test 1	Fail row	Failed test

Bits 14,15 contain the test that failed where the value is the following:

1— failed all zeros test
2— failed all ones test
3— failed alternating test

Bits 6-13 contain the row of the page that failed.
Bits 0-5 contain the failure code for the six RAM ICs on the board. Bits 0-4 contain the failure code for the RAMs for pod 1-5, and bit 5 contains the failure code for the RAM used for instructions. A one in the bit position indicates that that RAM provided incorrect information.

Address Counter Test

The Address Counter Test contains four subtests that check the functionality of the column and row address counters for module RAM. The four subtests use each of the four loop registers to perform the test.

The first step of the test is to load memory using the current loop register with a specific pattern for the address counter. Memory is loaded with 0x0000 except at predetermined RAM row and column positions, which are loaded with 0xFFFF.

The current loop register is used to set the address for the 0xFFFF loading. The loop register is also used to reset the addresses back to zero for starting the stepping process.

After memory has been loaded the clock is stepped through all possible RAM addresses checking for the correct data at each address.

Passing the Address Counter Test implies that each RAM memory location can be accessed by the RAM addressing circuitry while under control of the clocking circuit. Passing the test also implies that the loop registers are operating correctly.

DIAGNOSTIC INTEGER VALUE:

This test checks the counters of the entire module so the diagnostic integer will be valid for the master card and all expander-configured cards. The returned integer for a particular card has the following format:

BIT #:	15	14, 13, 12, 11, 10, 9, 8, 7, 6	5, 4, 3, 2, 1, 0
	Row/Col	Fail row	Failed RAM

Bit 15 is used to flag where the value of the fail row bits (6-14) came from. If the failing row value was less than 511 bit 15 is set to zero. If the failing row was greater than 511 bit 15 is set to one. The failed row bits (6-14) contain a value from 0 to 255.

Bits 0-5 contain the failure code for the six RAMs on the board. Bits 0-4 contain the failure code for the RAM for pods 1-5, and bit 5 contains the failure code for the RAM used for instructions. A one in the bit position indicates that that ram provided incorrect information.

Instruction Tests

This test contains three subtests that have unique descriptions. Each subtest is described below.

Passing the Instruction Tests implies that CPU addressing, RAM addressing, and the instruction decoder of the module responds properly to user commands.

SUBTEST #1 — Instruction Interface Test

This test checks the functionality of the break command in the instruction memory and the status register that reads the break.

On the first pass of this test, instruction memory is loaded with zeros (NOP). The module is run and the main status register polled to see that the hardware is running. If the hardware is stopped the test fails.

The second pass of the test places the break instruction on the next to last vector in memory. Again the hardware is started and the status is read. This time the module should stop or the test fails.

DIAGNOSTIC INTEGER VALUE:

This test is only valid for signals on the master board of the configuration. The values returned from any expansion cards will be zero. The integer returned will have the following bit format:

BIT #:	15 ,14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4	3, 2, 1, 0
	unused	Test Mode

The Test Mode bit positions have the following meaning:

0 — passed
1 — stopped without break
2 — failed to stop from software
4 — failed to stop with break
8 — stopped by something other than break

SUBTEST #2 — Wait Instruction Test

This test checks the functionality of the wait command in the instruction memory and the status register that reads the wait. The test is performed on each of the four event registers. The current wait event being tested is loaded on the first page of memory and a break instruction is loaded on the third page of memory.

The first pass of the test places a wait on no event in the event register. The hardware is started and a wait is begun for the vectors to hit the break instruction on the third page. The main status is checked to see that the hardware is stopped by the break instruction.

The second pass of the test places a wait on any condition in the event register. Again the hardware is started wait is begun. On this pass the hardware should be stopped by the wait condition and not by the break condition.

The final pass clears the current wait condition using the hardware wait clear command. The module should run from the current wait to the break condition and once again stop.

DIAGNOSTIC INTEGER VALUE:

This test is only valid for signals on the master board of the configuration. The values returned from any expansion cards will be zero. The integer returned will have the following bit format:

BIT #:	15, 14,13, 12	11, 10, 9, 8	7, 6, 5, 4	3, 2, 1, 0
	Event D	Event C	Event B	Event A

Each nibble corresponds to the event register being tested. The value of the nibble for the event register has the following definition:

0 — passed
1 — failed to stop on break with no event wait
2 — stopped on wait with setting of no event
3 — failed to stop on break or wait with wait any event
4 — failed to stop on wait with wait any event
5 — failed to clear wait
6 — stopped on false break condition

It should be noted that the value returned will be the last error encountered.

SUBTEST #3 — If Instruction Test

This test checks the functionality of the if branching.

Instruction memory is loaded with a wait on event 'a' instruction in the non-if branch of memory and a break instruction in the if branch.

The first pass of the test sets the branch pattern to a never branch condition. The module is started and a wait is begun for the vectors to get to the wait instruction. The hardware should stop on the wait instruction, not the break. The main status is checked to verify this stop condition.

The second pass of the test sets the branch pattern to always branch. Again the module is started and a wait is begun. In this case the break instruction should be the stop condition.

DIAGNOSTIC INTEGER VALUE:

This test is only valid for signals on the master board of the configuration. The values returned from any expansion cards will be zero. The integer returned will have the following bit format:

BIT #:	15 ,14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4	3, 2, 1, 0
	unused	Test Mode

The Test Mode bit positions have the following meaning:

0 — passed
1 — failed to stop on wait in non-if branch
2 — took if branch on no branch event
4 — failed to stop on break in if branch
8 — took non-if branch on any branch event

Output Patterns for testing with an external logic analyzer or oscilloscope

The performance test will set up two predefined patterns for examining the module from an external analyzer or scope. This allows the user to check the output pipeline for functionality and also helps to perform a quick check of all bit locations in the data VRAMS.

The data is output based on the frequency mode chosen by the user:

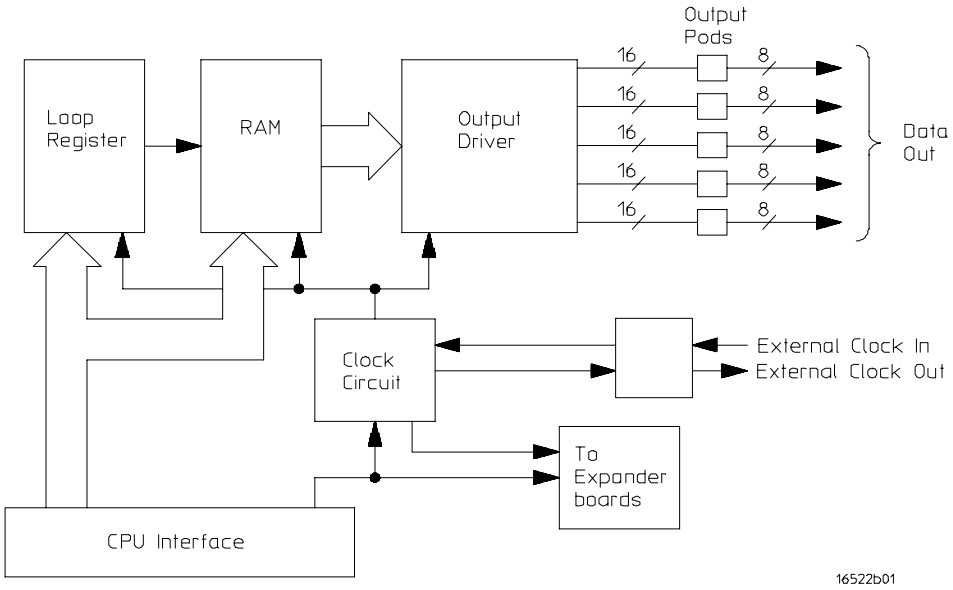
50MHz Mode — 20 ns period

100MHz Mode — 10 ns period

200MHz Mode — 5 ns period

Either a checkerboard pattern (alternating 1s and 0s across the output channels) or a walking 1s pattern are available.

Theory of Operation



Loop Register

The loop register holds the programmable vector flow information. When the module reaches the end of the vector listing, the loop register is queried for the RAM address location of the next user-programmed vector. In many cases, the next vector address location would be the start of the vector listing. Consequently the vectors would continue to loop from the end of the listing back to the beginning until the user instructs the module to stop.

RAM

Consisting of six 256Kx16 VRAM ICs and RAM addressing circuitry, the RAM stores the desired patterns that appear at the module output. The RAM addressing circuitry is merely a counter which addresses the pattern locations in RAM. When the end of the vector listing is reached, the addressing circuitry is loaded from the loop register with the address of the first vector of the listing to provide an uninterrupted vector loop. The RAM output is sent to the Output Driver circuit where the patterns are presented into a logic configuration usable by the output pods.

Output Driver

The output driver circuit is made up of a series of latch/logic translators and multiplexers. The latch/translators convert the working-level TTL signals to output-level ECL signals for each channel. The ECL-level signals are then directed to the multiplexers.

The multiplexers, one per channel, direct the programmed data patterns to the output channels. The single-ended ECL-level signals are converted to differential signals which are routed to the output cables and to the pods. Note that the differential ECL output signal of the pattern generator module is not suitable to directly drive ECL circuitry.

Clock Circuit

The clock circuit paces the loop register, the RAM address circuitry, and the multiplexers in the output driver according to the desired data rate. A 200 MHz clock source is directed through a divider circuit which provides a 100MHz and 50MHz clock in addition to 200MHz. The 200MHz, 100MHz, 50MHz and external clock signals are routed to a clock select multiplexer. The output of the multiplexer, which represents the user-selected clocking rate, is distributed to the above listed subcircuits on both the master board and all expander boards that are configured with the master board.

The output of the clock select multiplexer is also distributed to an external clock out circuit. The clock signal is routed to a bank of external clock delays, and then to an external clock delay select multiplexer. The output of this multiplexer, which represents the desired clock delay, is directed to the external clock out pin on the clock pod. Consequently, either the internal clock or external clock is redirected to the clock out pin on the clock pod with a user-selected clock delay.

CPU Interface

The CPU interface is a single programmable-logic device which interprets the 16500B/C Logic Analysis System backplane logic and translates the logic into signals to drive and program the pattern generator module.

Pod

The Clock or Data Pod converts the differential output ECL signal to the logic levels of interest. Because the output of the pattern generator module cannot directly drive ECL circuitry, the Clock and Data Pod is required to interface the pattern generator with the target system.

To replace the board and cable

To remove the cables

1. Remove power from the mainframe.
2. Remove the Agilent 16522A module from the mainframe. Refer to "Installation," in chapter 5.
3. Remove the three screws that secure the cable clamp to the rear panel.
4. If you are replacing a single cable, unplug the cable and continue with "To install the cables".
5. If you are replacing the circuit board, unplug all cables from the board. Continue with "To replace the circuit board."

To replace the circuit board

1. Remove the cables (use procedure above).
2. Remove four screws attaching the ground spring and rear panel to the circuit board, then remove the back panel and the ground spring.
3. Replace the faulty circuit board with a new circuit board. On the circuit board, make sure the 20-pin ribbon cable is connected between J7 and J8.
4. Position the ground spring and back panel on the back edge of the replacement circuit board. Install four screws to connect the back panel and ground spring to the circuit board.
5. Continue with "To install the cables"

To install the cables

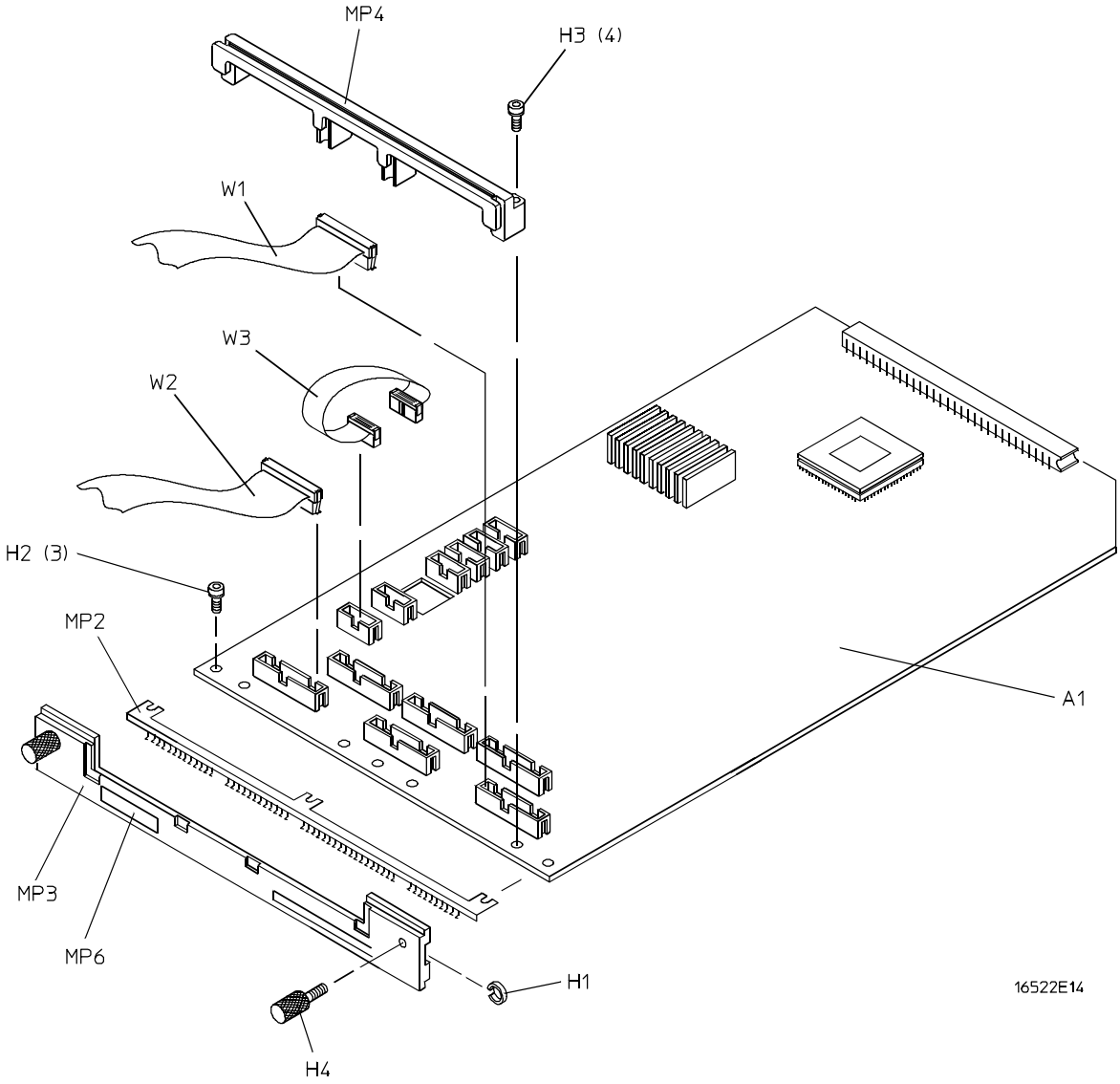
1. Plug the five data cables and the clock cable in the appropriate connectors on the circuit board. If a single cable is being replaced, plug the cable in the appropriate connector.

   Note that the clock cable is connected to J6 on the Agilent 16522A board.
2. Position the cables on the rear panel so the cable clamp can be properly installed on the rear panel.
3. Install three screws that secure the cable clamp onto the rear panel.
4. Install the Agilent 16522A module into the mainframe. Refer to "Installation" in chapter 5.

Agilent 16522A Replacement Parts

Agilent 16522A Replaceable Parts

Ref. Des	Agilent Part Number	Qty	Description
A1	16522-69501		Rebuilt circuit board
A1	16522-66501	1	Circuit board assembly
H1	0510-0684	2	Retaining ring
H2	0515-0430	3	Machine screw
H3	0515-0665	4	MSPH M3X14 T10
H4	16500-22401	2	Rear panel screw
MP1	16500-41201	6	Ribbon cable ID clip
MP2	16500-29101	1	Ground spring
MP3	16510-40501	1	Rear panel
MP4	16510-40502	1	Cable Clamp
MP5	01650-94309	1	Probe label
MP6	16522-94301	1	ID label
W1	16522-61601	5	Data cable (J1 - J5)
W2	16522-61602	1	Clock cable (J6)
W3	16522-61603	1	Interconnect cable



Exploded View of Agilent 16522A

To clean the pattern generator module

With the instrument turned off and unplugged, use mild soap and water to clean the front of the module. Harsh soap might damage the water-based paint.

5

Installation

Preparing For Use

This section gives you instructions for preparing the pattern generator module for use.

Power Requirements

All power supplies required for operating the pattern generator are supplied through the backplane connector in the mainframe.

Operating Environment

The pattern generator module's reliability is enhanced when operating the module within the following ranges:

- Temperature: +20 °C to +35 °C (+68 °F to +95 °F)
- Humidity: 20% to 80% noncondensing

Storage

Store or ship the pattern generator in environments within the following limits:

- Temperature: –40 °C to + 75 °C
- Humidity: Up to 90% at 65 °C
- Altitude: Up to 15,300 meters (50,000 feet)

Protect the module from temperature extremes which cause condensation on the instrument.

To inspect the module

1 Inspect the shipping container for damage.

If the shipping container or cushioning material is damaged, keep them until you have checked the contents of the shipment and checked the instrument mechanically and electrically.

2 Check the supplied accessories.

Part Number	Description	Qty
16500-41201	Ribbon cable ID clip	6
16522-61601	Data output cable	5
16522-61602	Clock cable	1
16522-61603	Interconnect cable	1

3 Inspect the product for physical damage.

Check the module and the supplied accessories for obvious physical or mechanical defects. If you find any defects, contact your nearest Agilent Technologies Sales Office. Arrangements for repair or replacement are made, at Agilent Technologies' option, without waiting for a claim settlement.

To prepare the mainframe

CAUTION Turn off the mainframe power before removing, replacing, or installing the module.

CAUTION Electrostatic discharge can damage electronic components. Use grounded wrist straps and mats when performing any service to this module.

1 Turn off the mainframe power switch, then unplug the power cord. Disconnect any input or output connections.

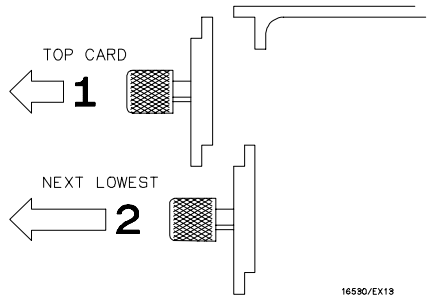
2 Plan your module configuration.

If you are installing a one-card module, use any available slot in the mainframe. If you are installing a multi-card module, use adjacent slots in the mainframe.

3 Loosen the thumb screws.

Cards or filler panels below the slots intended for installation do not have to be removed.

Starting from the top, loosen the thumb screws on filler panels and cards that need to be moved.



4 Starting from the top, pull the cards and filler panels that need to be moved halfway out.

CAUTION

All multi-card modules will be cabled together. Pull these cards out together to prevent damage to the cables and connectors.

5 Remove the cards and filler panels.

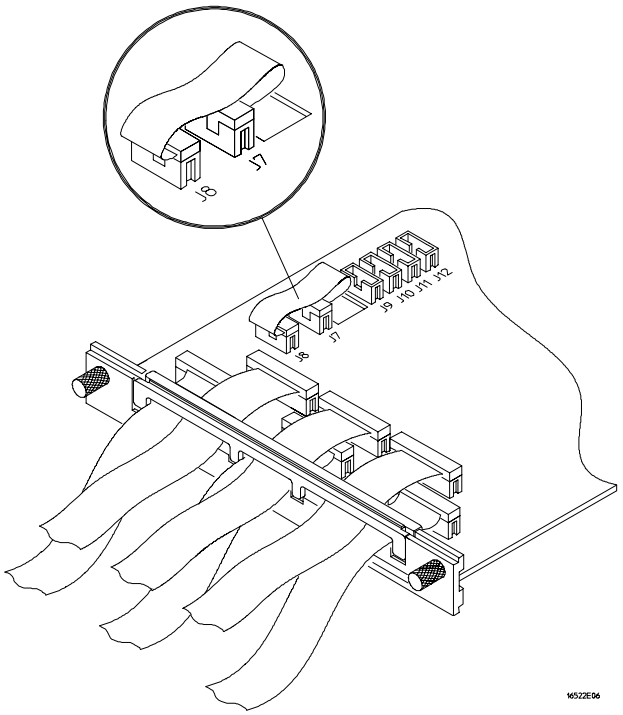
Remove the cards or filler panels that are in the slots intended for the module installation. Push all other cards into the card cage, but not completely in. This is to get them out of the way for installing the module.

To configure a one-card module

- When shipped, each module is configured as a master in a one-card module. The cables on each module should be connected as shown in the figure.
- To configure a multi-card module into one-card modules, remove the cables connecting the cards. then connect the free end of the 2x10 cable to the connector labeled "Master Output" (J8) on each card (see figure below).

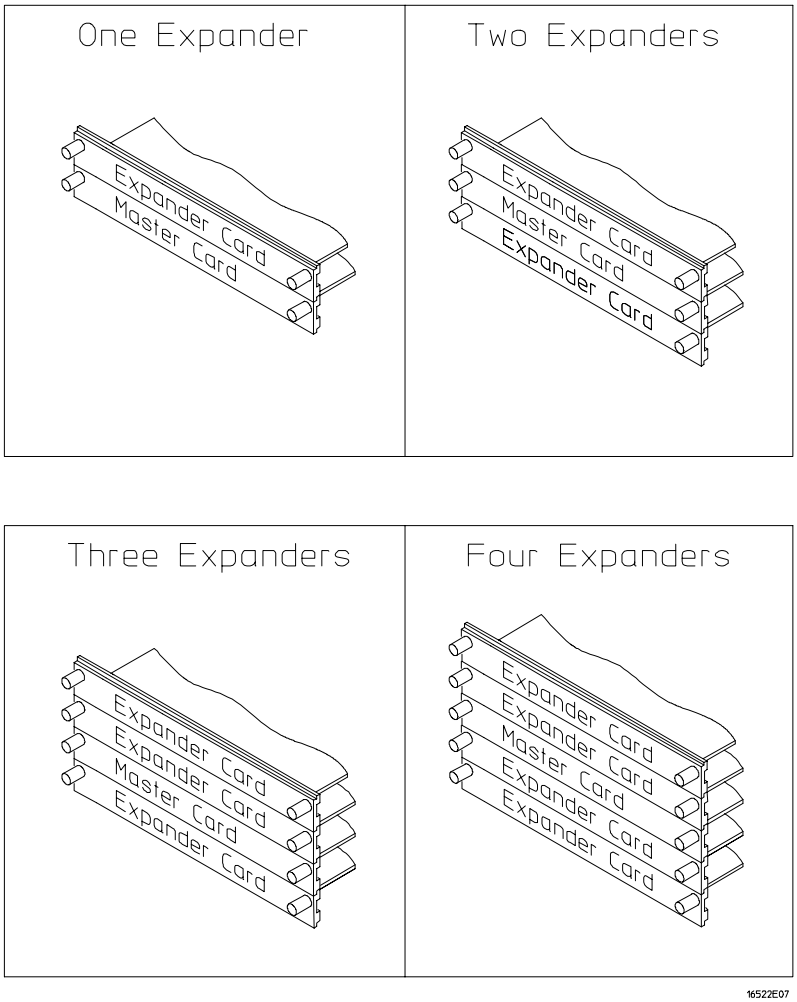
CAUTION

If you pull on the flexible ribbon part of the 2x10 cable, you might damage the cable assembly. Using your thumb and finger, grasp the ends of the cable connector. Apply pressure to the ends of the cable connector to disengage the metal locking tabs of the connector from the cable socket on the board. Then pull the connector from the cable socket.



To configure a multi-card module

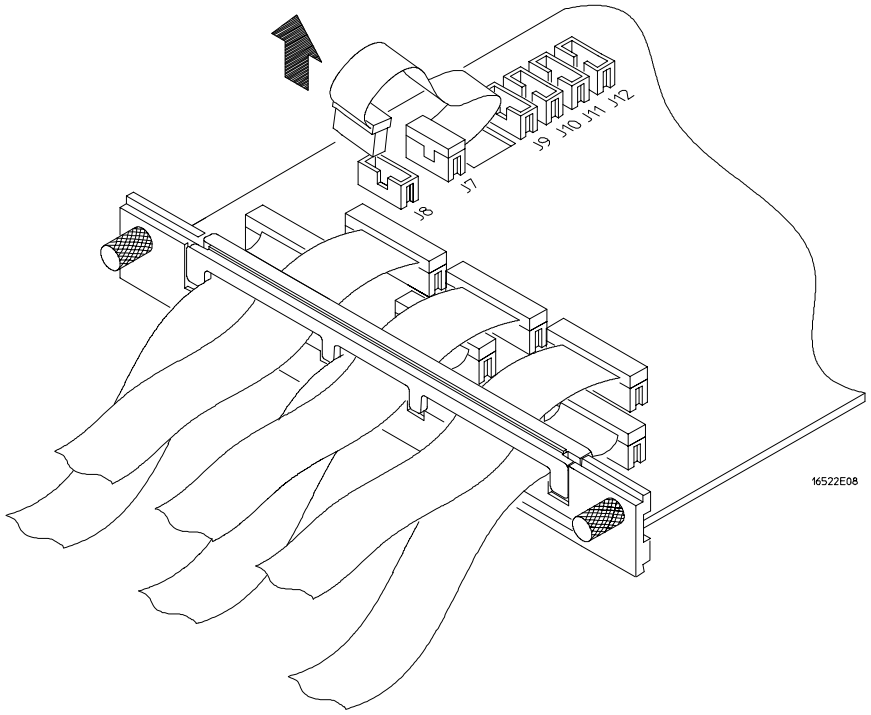
1 Plan the configuration. Multi-card modules can only be connected as shown in the illustration. Select the card that will be the master card, and set the remaining boards aside.



2 On the expander card, disconnect the end of the 2x10 cable that is plugged into the connector labeled "Master Output."

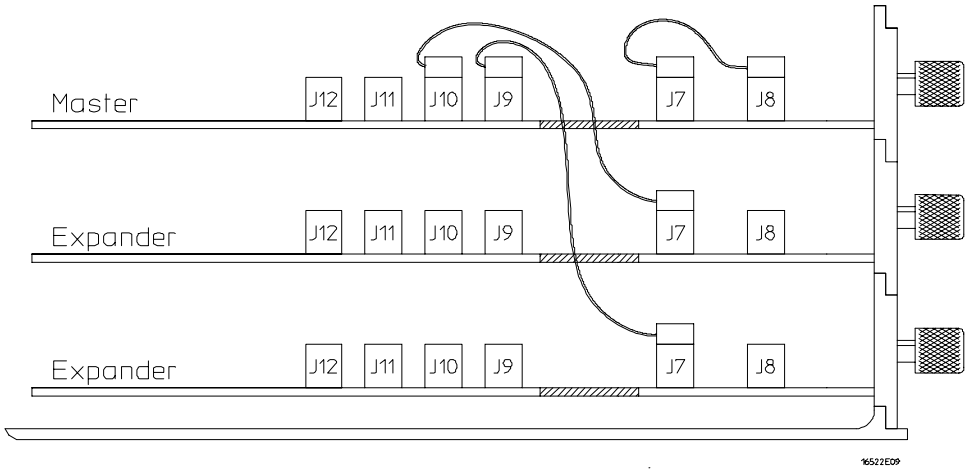
CAUTION If you pull on the flexible ribbon part of the 2x10 cable, you might damage the cable assembly.

Using your thumb and finger, grasp the ends of the cable connector. Apply pressure to the ends of the cable connector to disengage the metal locking tabs of the connector from the cable socket on the board. Then, pull the connector from the cable socket.

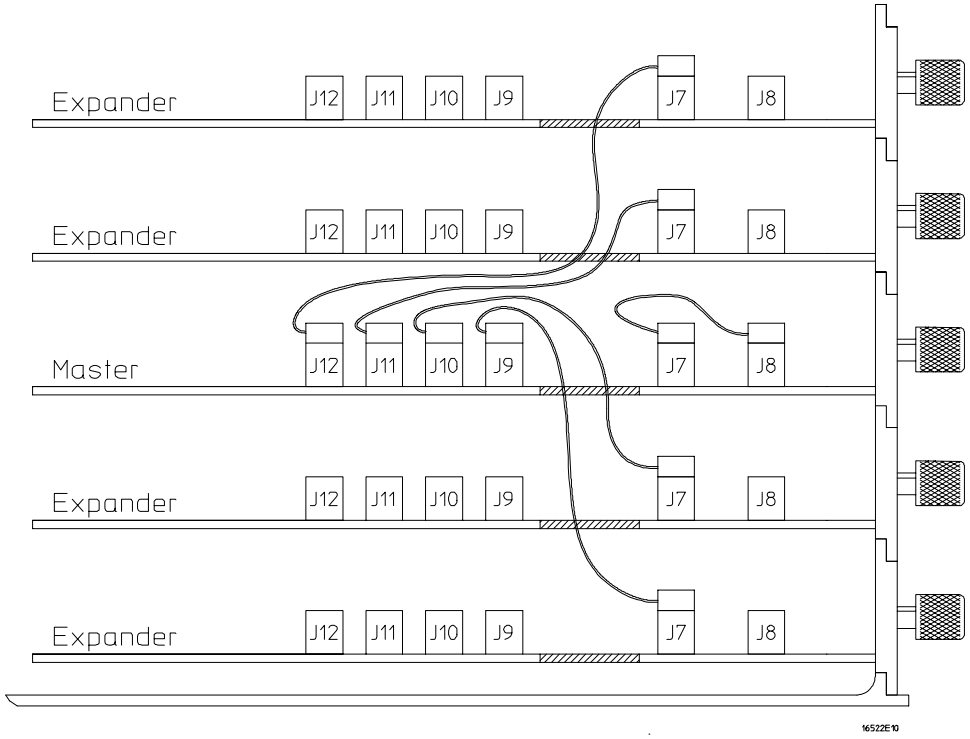


3 Place the master card on top of any expander cards that are under the master card. Feed the free end of the 2x10 cables of the expander cards through the cable access holes to the master card. Plug the 2x10 cables into J9 (bottom-most expander in a five-card configuration) and J10 (expander that is next to the master card) on the master card.

The illustration below shows the bottom three cards of a five-card configuration.



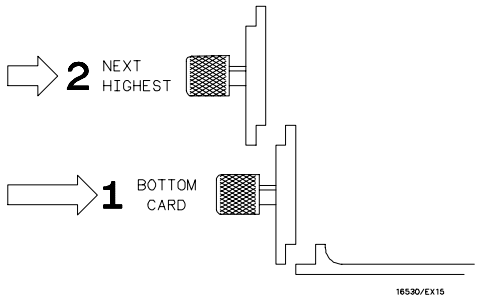
4 Place the remaining expander cards on top of the master board. Feed the free end of the 2x10 cables of the expander cards through the access holes to the master card. Plug the 2x10 cables into J11 (expander that is next to the master card) and J12 (top-most expander in a four or five-card configuration) on the expander cards.



To install modules

1. Slide the cards which are above the slots into which you want to put modules about halfway out of the mainframe.
2. With the probe cables facing away from the instrument, slide the modules approximately halfway into the mainframe.
3. Slide the assembled modules into the mainframe, but not completely in.

   Each card in the instrument will be firmly seated and tightened in step 5.
4. Position all cards and filler panels so that the endplates overlap.



5. Seat the cards and tighten the thumbscrews.

   Starting with the bottom card, firmly seat the cards into the backplane connector of the mainframe. Keep applying pressure to the center of the card endplate while tightening the thumbscrews finger-tight. Repeat this for all cards and filler panels from the bottom up.

CAUTION Correct air circulation keeps the instrument from overheating. For correct air circulation, filler panels must be installed in all unused card slots. Keep any extra filler panels for future use.

6

Loading ASCII files

Introduction

The user may create pattern generator files and load them as ASCII files via one of the remote communication interfaces or by loading an ASCII disk file.

Regardless of the load method selected, the general format of the file must conform to certain guidelines. In general, an ASCII file consists of a block of setup information (clock specs, labels, etc.) followed by a block of pattern generator data.

There are a few minor differences between the format required for a communication download vs. that of a disk file load.

Disk files are loaded into the pattern generator module via the LOAD command on the disk menu.

Some general restrictions are:

- Data is assumed to be entirely hexadecimal base.
- No instructions are allowed in the data.
- No macros are defined or invoked in the data.
- All labels consist of adjacent bits.

A special "`ASCII  000000`" string is required to uniquely identify an ASCII disk file. This string cannot be used when loading ASCII files via one of the remote communication interfaces. This line must be the first line of the disk file. It consists of 5 blanks and 6 zeros.

ASCII file commands

In addition to the unique ASCII file commands described here, the user may wish to include some standard FORMat commands in the ASCII file, such as those that are used to specify the clock or output mode.

The only FORMat commands that are permitted are FORMat: MODe, CLOCk, and DELay. Refer to Chapter 3 for the syntax of these commands.

Note that there is no section header prefix for these ASCII file command strings.

Refer to the example programs at the end of this chapter for usage examples of the various commands described in this chapter.

ASCDown Command

Command

ASCDown

The ASCdown command is used to signal the start of an ASCII file load. It causes the current pattern generator label and sequence structures to be cleared and reset to a default state.

The ASCDown command must precede any label definitions and the data portion of an ASCII file load.

Example

```
ASCDown
```

LABel

Command `LABel <name_str>,<width>`

`<name_str>` label string up to six characters in length.

`<width>` integer number of bits in the label (1 through 32).

The LABel command is a special means of specifying labels for use by an ASCII file. The label bits are assigned from most to least significant bits across the output pods. Labels may only contain adjacent bits. The user must specify the label string and the width of the field. The label base is hexadecimal.

There is a maximum of 126 labels. No label may be more than 32 bits wide. If a label is too wide (too many bits) for the remaining unused pattern generator bits, it will be discarded. Use of the FORMat:LABel command after the ASCDown command will generate an error.

Example

```
LABel 'TEST1',7

LABel 'ADDR',13
```

VECTor

Command `VECTor <char_count>`

`<char_count>` a ten character string starting with a '#8' and including the total file size count.

The VECTor command is used after the end of the header/setup commands to signal the start of the actual pattern generator data in an ASCII file.

The VECTor command is used with a parameter that specifies the exact byte count of the data block. This count must include all data characters, all blank characters, and all line termination (DOS `cr/lf` or UNIX `cr`) characters. The file character count is the sole criteria used to determine when the bus file transfer is complete. If a disk file is used, the character count has no meaning and can be any value or deleted from the command string.

If the file character count does not match the actual data byte count of the file an error condition will occur. If the actual data count exceeds the byte count passed in with the VECTor command, excess data will be lost (and treated as remote control bus command(s)). If the actual data count is less than the data count passed in with the VECTor command, the bus transfer will appear to hang while the Agilent 16500B/C system waits for the 'remaining' data. The controller sending the file may, or may not, time-out and terminate the bus transfer. Generally, recovery from this condition involves sending more data until the data byte count is satisfied.

The file character count is contained in a string with a specific format. The actual count is right justified in a ten-character string that starts with a '#8' followed by eight digits. These ten characters are NOT part of the file character count.

The following page shows an example of this.

No data is allowed in the same line as the VECTor command. The line termination in the VECTor command line IS included in the character count for the file.

The <char_count> field is not required as part of the VECTor command when creating a disk file, and will be ignored if included.

Example:

```
VECTor #800010457
```

Vector Data

The data portion of the ASCII file is basically an array of hexadecimal data fields. Each row of the array corresponds to a single line of the main program. Each column of the array corresponds to a single label as defined on the format menu.

Data fields are separated by one or more blank characters. A line termination (carriage return or carriage return + line feed) signals the end of a line and the start of a new line. If a data field has more data than the label width would indicate, only the least significant bits of the data field are used. If there are more data fields in a row than there are labels, the extra data fields (last data fields in the row) are ignored. If there are fewer data fields in a row than there are labels, the data for the extra (right-most) labels will be zero.

Data lines consisting of only line termination characters are ignored.

The MAIN SEQUENCE must have at least two data lines. In the ASCII data file, a row consisting of only '*M' is used to signal the start of the main sequence. If there is to be no data in the init sequence, the first row of the file after the VECTor command must be '*M'. Note that the quotes in '*M' are not really in the file. The line termination after the '*M' (as well as the '*M') must be included in the character count for a file loaded via a remote bus.

Any characters that are not valid hexadecimal digits (i.e.: 0 through 9, or upper/lower case a through f) are ignored and treated as field separators. This could cause problems if a typo appears in the middle of a data value (for example, '12R4' will be assigned to two labels as '12' and '4').

The last data row of the file must end with a line termination as this is the flag to load the data row into the data structure. Failure to do this will result in a short main program.

When counting file characters be aware of how a particular file generator (editor) terminates a line. DOS-based systems use two characters. Be sure to account for line termination character(s) in the overall file character count.

The ASCII file load mechanism assumes correctness in the data file and any header commands. Error handling is rather basic and treating unexpected characters as field separators could create bizarre results when parsing the file. Error messages point to the line number where the parser thinks the error occurred, but the line count may not be exact because of parsing problems with the data.

When using a LAN interface to send ASCII data, an extra line feed <lf> is required at the end of the file. This <lf> IS NOT included in the <char_count> value. It is required to ensure the data buffer is flushed.

Serious problems will cause the default main program to be loaded in an effort to avoid locking up the Agilent 16500B/C system.

FORMat:xxx

Command

```
FORMat:MODE
FORMat:CLOCK
FORMat:DELay
```

These commands transfer set fields from the Format menu. The existing clock scheme is used if nothing is specified here. Command syntax is same as normal bus commands.

See also

Full command description in Chapter 3.

Examples

```
FORMat:MODE FULL
FORMat:CLOCk INT,5E-9
FORMat:DELay 1E-9
```

Loading an ASCII file over a bus (example)

To load an ASCII file over the bus use the following example. A few items to be noted:

- Line numbers are added for documentation only and are NOT part of the actual remote bus commands.
- In this example, the string `'<lf>'` is a generic line feed sequence and counts as a single character.

```
010  SELect <slot><lf>

020  ASCDOWN<lf>

030  FORM:MODE FULL<lf>

040  LABEL 'LAB1',8<lf>
041  LABEL 'DATA',8<lf>
042  LABEL 'TEST',9<lf>
043  LABEL 'CLK',3<lf>
044  LABEL 'BIG',12<lf>

050  VECT #800000092<lf>
060  12 34 56 7 89A<lf>
070  0 22 7 0 FFF<lf>
080  A0 33 00 1 111<lf>
090  *M<lf>
100  92 6F 00 1 FF0<lf>
110  CA CA 00 1 00F<lf>
120  00 10 11 0 ABC<lf>
```

Notes

- Lines 010 through 044 can be sent as discrete remote control commands or included in a single file (with the data) and loaded via the bus.
- Other format commands could be used in place of or in addition to line 030.
- The label sequence seen in lines 040 through 044 will result in a specific bit assignment (see below). A different ordering of the LABel commands would give a different ordering to the bits.

- There is a space before the '#8' in line 050
- The character count in line 050 is based on:
  - 15 characters (10 digits, 4 blanks, 1 <lf>) each in lines 060, 080, 100, 110, and 120.
  - 3 characters in line 090
  - 13 characters in line 070
  - 1 character (the <lf>) in line 050

The previous file when loaded from GPIB creates the following format and sequence set ups when loaded into slot C.

Format Specification

Clock Source: Internal

Vector Output Mode: Full Channel 100 Mbit/s

Clock Period: 10 ns

Clock Out Delay Setting: 0

		Pod C5	Pod C4	Pod C3	Pod C2	Pod C1
Label	**Pol**	7.....0	7.....0	7.....0	7.....0	7.....0
LAB1	+	********	.....	.....	.....	.....
DATA	+	.....	********	.....	.....	.....
TEST	+	.....	.....	********	*.....	.....
CLK	+	.....	.....	.....	.***....	.....
BIG	+	.....	.....	.....	....****	********

Sequence Listing

Line	Label > Base >	LAB1 Hex	DATA Hex	TEST Hex	CLK Hex	BIG Hex
0	INIT SEQUENCE START					
1		12	34	56	7	89A
2		00	22	07	0	FFF
3		A0	33	00	1	111
4	INIT SEQUENCE END					
5	MAIN SEQUENCE START					
6		92	6F	00	1	FF0
7		CA	CA	00	1	00F
8		00	10	11	0	ABC
9	MAIN SEQUENCE END					

Loading an ASCII file from a disk (example)

To load an ASCII file from a disk use the following example. A few items to be noted:

- Line numbers are added for documentation only and are NOT part of the actual disk file.
- In this example, the string "<lf>" is a generic line feed sequence and counts as a single character.

```
010  ASCII     000000<lf>

020  ASCDOWN<lf>

030  FORM:MODE FULL<lf>

040  LABEL LAB1,8<lf>
041  LABEL DATA,8<lf>
042  LABEL TEST,9<lf>
043  LABEL CLK,3<lf>
044  LABEL BIG,12<lf>
050  VECT<lf>
060  12 34 56 7 89A<lf>
070  0 22 7 0 FFF<lf>
080  A0 33 00 1 111<lf>
090  *M<lf>
100  92 6F 00 1 FF0<lf>
110  CA CA 00 1 00F<lf>
120  00 10 11 0 ABC<lf>
```

Notes

- Line 010 is the disk file header and must be the first 16 characters in the file. The string 'ASCII 000000' is 16 characters long with five spaces and six zeros.

- Other format commands could be used in place of or in addition to line 030.
- The label sequence seen in lines 040 through 044 will result in a specific bit assignment (see above). A different ordering of the LABel commands would give a different ordering to the bits.
- Note that a character count is not specified on line 050.

Index

Document Warranty

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

Agilent Technologies shall not be liable for errors contained herein or for damages in connection with the furnishing, performance, or use of this material.

Safety

This apparatus has been designed and tested in accordance with IEC Publication 348, Safety Requirements for Measuring Apparatus, and has been supplied in a safe condition. This is a Safety Class I instrument (provided with terminal for protective earthing). Before applying power, verify that the correct safety precautions are taken (see the following warnings). In addition, note the external markings on the instrument that are described under "Safety Symbols."

Warning

- Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.
- Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock of fire hazard.
- Service instructions are for trained service personnel. To avoid dangerous electric shock, do not perform any service unless qualified to do so. Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.
- If you energize this instrument by an auto transformer (for voltage reduction), make sure the common terminal is connected to the earth terminal of the power source.
- Whenever it is likely that the ground protection is impaired, you must make the instrument inoperative and secure it against any unintended operation.
- Do not operate the instrument in the presence of flammable gasses or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.
- Do not install substitute parts or perform any unauthorized modification to the instrument.
- Capacitors inside the instrument may retain a charge even if the instrument is disconnected from its source of supply.
- Use caution when exposing or handling the CRT. Handling or replacing the CRT shall be done only by qualified maintenance personnel.

Safety Symbols

Instruction manual symbol: the product is marked with this symbol when it is necessary for you to refer to the instruction manual in order to protect against damage to the product.

Hazardous voltage symbol.

Earth terminal symbol: Used to indicate a circuit common connected to grounded chassis.

WARNING

The Warning sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a Warning sign until the indicated conditions are fully understood and met.

CAUTION

The Caution sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a Caution symbol until the indicated conditions are fully understood or met.

Agilent Technologies
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901

Product Warranty

This Agilent Technologies product has a warranty against defects in material and workmanship for a period of one year from date of shipment. Some newly manufactured Agilent Technologies products may contain remanufactured parts which are equivalent to new in performance. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies.

For products returned to Agilent Technologies for warranty service, the Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of merchantability or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the buyer's sole and exclusive remedies. Agilent Technologies shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales Office.

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

About this edition

This is the *Agilent 16522A 200-M Vectors/s Pattern Generator User's Guide.*

Publication number
16522-97007, June 2000
Printed in USA.

Print history is as follows:
16522-97006, February 1999
16522-97004, December 1996
16522-97005, August 1997

New editions are complete revisions of the manual. Many product updates do not require manual changes and manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

DECLARATION OF CONFORMITY

according to ISO/IEC Guide 22 and EN 45014

Manufacturer's Name: Agilent Technologies

Manufacturer's Address: Colorado Springs Division
1900 Garden of the Gods Road
Colorado Springs, CO 80907 USA

declares, that the product

Product Name: Pattern Generator Module

Model Number(s): Agilent 16522A

Product Option(s): All

conforms to the following Product Specifications:

Safety:	IEC 1010-1:1990+A1 / EN 61010-1:1993	
	UL 3111	
	CSA-C22.2 No. 1010.1:1993	
EMC:	CISPR 11:1990 / EN 55011:1991	Group 1 Class A
	IEC 555-2:1982 + A1:1985 / EN 60555-2:1987	
	IEC 555-3:1982 + A1:1990 / EN 60555-3:1987 + A1:1991	
	IEC 801-2:1991 / EN 50082-1:1992	4 kV CD, 8 kV AD
	IEC 801-3:1984 / EN 50082-1:1992	3 V/m, {1kHz 80% AM, 27-1000 MHz}
	IEC 801-4:1988 / EN 50082-1:1992	0.5 kV Sig. Lines, 1 kV Power Lines

Supplementary Information:

The product herewith complies with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC and carries the CE marking accordingly.

This product was tested in a typical configuration with Agilent Technologies test systems.

Colorado Springs, 04/03/95

John Strathman, Quality Manager

European Contact: Your local Agilent Technologies Sales and Service Office or Agilent Technologies GmbH, Department ZQ / Standards Europe, Herrenberger Strasse 130, D-71034 Böblingen Germany (FAX: +49-7031-14-3143)

Product Regulations

Safety

IEC 1010-1:1990+A1 / EN 61010-1:1993
UL 3111
CSA-C22.2 No.1010.1:1993

EMC

CE
ISM 1-A

This Product meets the requirement of the European Communities (EC) EMC Directive 89/336/EEC.

Emissions EN55011/CISPR 11 (ISM, Group 1, Class A equipment)

Immunity

EN50082-1	Code[1]	Notes[2]
IEC 555-2	1	
IEC 555-3	1	
IEC 801-2 (ESD) 4kV CD, 8kV AD	1,2	*
IEC 801-3 (Rad.) 3 V/m	2	
IEC 801-4 (EFT) 0.5 kV, 1kV	1,2	*

[1] Performance Codes:
1 PASS - Normal operation, no effect.
2 PASS - Temporary degradation, self recoverable.
3 PASS - Temporary degradation, operator intervention required.
4 FAIL - Not recoverable, component damage.

[2] Notes: (none)

Sound Pressure Level N/A